National Aeronautics and
Space Administration

Lyndon B. Johnson Space Center
Houston, Texas 77058

JSC-17471

SUBSATELLITE ORBITAL ANALYSIS PROGRAM (SOAP) USER'S GUIDE

Job Order 44-169

Prepared by

Lockheed Engineering and Management Services Company

for

EXPERIMENT SYSTEMS DIVISION

July 1981

(NASA-CR-161088) SUBSATELLITE ORBITAL ANALYSIS PROGRAM (SOAP) USER'S GUIDE (Lockheed Engineering and Management) 294 p HC A13/MF A01 CSCL 22A

N81-33220

Unclas G3/13 37891

JSC-17471

SUBSATELLITE ORBITAL ANALYSIS PROGRAM (SOAP) USER'S GUIDE

Job Order 44-169

PREPARED BY

K. G. Castle

J. M. Voss

J. S. Gibson

APPROVED BY

F. E. Volentine, Supervisor
Communications and Control Section

F. N. Barnes, Manager
Dynamic Systems Department

July 1981

LEMSCO-16583

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
JSC-17471		

4. Title and Subtitle	5. Report Date
Subsatellite Orbital Analysis Program (SOAP) User's Guide	July 1981
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
K. G. Castle, J. M. Voss, J. S. Gibson Lockheed Engineering and Management Services Co.	LEMSCO-16583
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
Lockheed Engineering and Management Services Co. 1830 NASA Road 1 Houston, Texas 77058	NAS 9-15800
	13. Type of Report and Period Covered

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Lyndon B. Johnson Space Center Houston, Texas 77058 Technical Monitor: R. H. Gerlach	

15. Supplementary Notes

16. Abstract

This document describes the features and use of the Subsatellite Operational Analysis Program. This simulation models several earth-orbiting vehicles, their pilots, control systems, and interaction with the environment. The use of the program, its input and output capabilities, its executive structures, and properties of the vehicles and environmental effects which it models are described.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Multivehicle Simulation Subsatellites Onorbit Operation Space Shuttle Operation	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	294	

*For sale by the National Technical Information Service, Springfield, Virginia 22161

CONTENTS

TABLES

FIGURES

SOAP USER'S GUIDE

1. INTRODUCTION

The Subsatellite Orbital Analysis Program (SOAP) is an all-digital functional simulation designed to model the interaction of several independent fliers in the on-orbit environment. Equations of motion are integrated for both rotation and translation of each vehicle. Each has its own pilot, control system, sensors, and environmental interactions. The properties of each vehicle and the fidelity with which each is modeled are determined by the user. The SOAP supplies a library of math models for various vehicles, a structure for setting up the relationships among various math models, and a mechanism for general input and output.

The SOAP is written in FORTRAN and is designed for use with a UNIVAC 1100 series computer. It differs from most FORTRAN simulations in that the order in which computations are made is determined not only by logical branching within subroutines or within a driver program, but on the basis of simulation time. The SOAP maintains an internal clock and also maintains a schedule of events. These events can include such procedures as execution of control system routines, sampling the vehicle dynamic state by sensors, response of control jets to a pilot's commands, the recording of vehicle states for output, or the pilot checking the status of a fuel gauge. The user need only develop subroutines modeling the desired events and specify their whereabouts to the SOAP executive program. Events which occur at regular time intervals are scheduled by alerting the SOAP executive. Routines which occur only irregularly or which occur only when some specified condition is met can be scheduled or called by other routines during the execution of the simulation. The advantage in using the SOAP executive lies in the fact that the sequence of events can be arranged to reflect realistic order and time of occurence. There is no need for the times at which events occur to be integer multiples of a single time unit. In addition, the SOAP automatically creates a driver to call all the subroutines desired and to produce appropriate input and output. This facilitates the setup and execution of new simulations.

From the user's point of view, each vehicle has a separate pilot, control system, dynamics, and sensors. Each vehicle and its pilot may or may not have knowledge of the activities of any other. Typically, communication between the control system, pilot, and dynamics takes place by issuing commands in the form of setting locations in COMMON blocks. Communication with the user typically takes the form of inputing into locations in COMMON blocks at the start of simulation, printing out information during the simulation (either at fixed time intervals or at the beginning and end), and of plots generated after the simulation.

Performing a simulation using the SOAP typically involves four stages of program execution:

a. Preprocessing - The math models to be used, the logical and/or temporal sequence of events, and the input/output desired are specified. A program is executed which sets up the driver program and writes several subroutines which act as subdrivers.
b. Mapping or Object Code Collecting - Instructions are provided which describe the location of all object code which will be needed by the simulation.
c. Simulation Time Proceeds - Input to the simulation is typically performed by specifying values for COMMON block locations at the start of the simulation. The simulation proceeds until it is terminated due to the attainment of some internal condition. Printed output can be obtained during this stage.
d. Postprocessing - Data which has been collected during the simulation is plotted.

From the user's point of view, steps a, b, and d are not of particular interest. They simply require a series of steps which must be followed in order to produce the simulation desired. They are, therefore, documented by a user's guide (section 2) and by brief descriptions of individual subroutines.

Step c, the simulation, is discussed further.

The SOAP system maintains a clock and a driver program. It takes care of input and output automatically, but most other activities are specified by the user. These routines, typically, are either scheduled or called by other

routines. The SOAP system is not aware of the names of the routines until the preprocessing stage.

Several activities have a special status within a SOAP simulation. One such activity is the process of computing the position and attitude of each vehicle. This activity requires that forces and torques due to such effects as gravity, gravity gradient, aerodynamic drag, and reaction control jets be computed. It also requires that the mass and the inertia tensor of each vehicle be updated. This series of computations is termed the "Environment" (of the Flight Control System and pilot). In most situations the entire dynamics computation is performed at once (for all vehicles). The modeling of each vehicle need not, however, be the same. Both the causes of force and torque and the way in which the causes are modeled can differ.

The Environment communicates with the rest of the simulation and the user via COMMON blocks. The Environment routines and COMMON blocks have a special status within the SOAP, because the SOAP executive knows their names. This knowledge of the names of the COMMON blocks involved is required to enable the user to input data from the outside. The knowledge of the names of models allows the SOAP executive to create a driver. It should be noted that any of these names can be changed, but a system change is required.

Figure 1-1 displays the organization of the Environment routines. The names of the routines displayed are really names of driver routines. Each driver does little except call the routines which perform the computation for each vehicle. The usual execution involves the driver program ACTVEH calling vehicle models for each vehicle (Space Shuttle Orbiter, Subsatellite, Manned Maneuvering Unit, etc.). These vehicle models typically call all of the force and torque computations involved with a single vehicle.

Two other special categories of model exist: "Passive Vehicles" which represents a free-flying vehicle without active control, and Sensors. Each of these categories is special to the SOAP executive in that the routine name and supporting COMMON block name is recognized. The Sensor block consists of a sensor control driver which, in turn, calls models for the sensors (e.g., gyros, horizon sensors) associated with each vehicle.

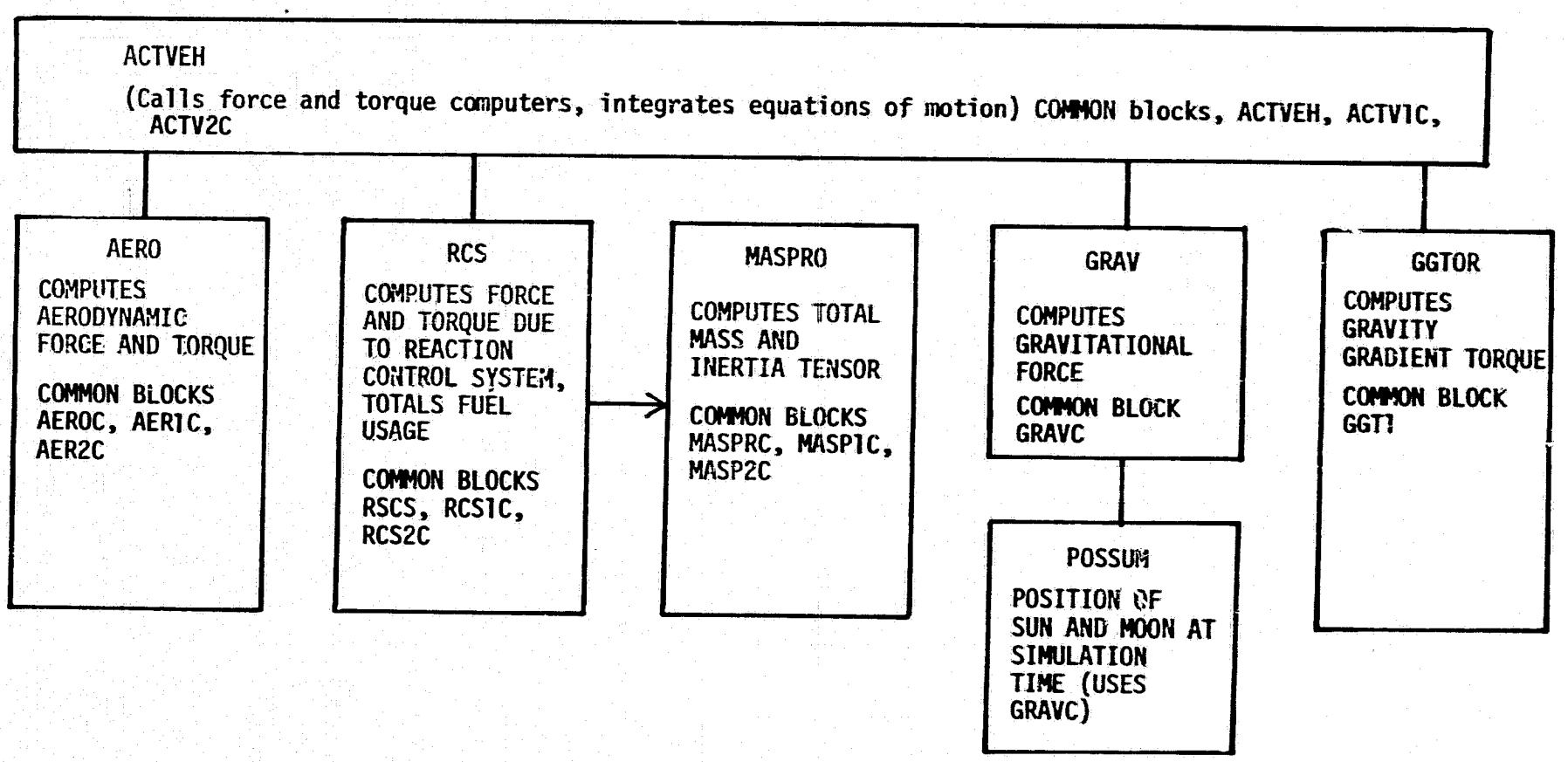


Figure 1-1.- SOAP Environment modeling.

All routines in the Environment, Sensors, or Passive Vehicles are specified to the Preprocessor by listing the version names of those drivers which are to be used to the Preprocessor (see section 2). All routines used, both drivers and actual models, must have the relevant object code collected by mapping instructions during the map execution (phase b above). The actual execution of these routines is triggered by the necessity of updating the dynamics (as requested from other parts of the simulation) or of carrying out a command (e.g., turn on jets) issued by the flight control or pilot. The dynamic state can also be updated at regular intervals by scheduling a driver program termed CALENV at the desired intervals.

The SOAP differs from previous similar simulation programs (notably the Space Shuttle Functional Simulator (SSFS)) in that the computations supporting the vehicle dynamics can be performed entirely in double precision.

All portions of the simulation not previously specified are specified by the user at the time of Preprocessing (step a) and/or mapping (step b).

Any operation desired by the user can be written as a subroutine, either callable by another routine or occurring on the basis of time or logical decision. Any routine which is not solely called by other routines is "scheduled." That is, the user tells the SOAP executive the name of a particular simulation subroutine, the simulation time at which it begins to be used, the time interval between executions, and the time when it ceases to be used. If any routine is to be executed before or after a scheduled routine but at the same simulation time, this can also be specified. The preprocessor uses this information to create a calling routine (SELECT) with knowledge of the routines to be called. The map instructions must include the whereabouts of object code for all routines used, whether scheduled or called out.

Some models, such as control jet models, accept commands from the pilot or control system and may even delay the execution of the action commanded. Such a delay would be specified by the environment model accepting the commands. The SOAP master clock ensures that the delay is applied. Table 1-I lists commands which are in use by some of the SOAP models. The SOAP system assigns the usage of these command indices at the time of simulation, so there is no necessity to avoid the command indices used by a model when the model is not in use.

TABLE 1-I.- SOAP ENVIRONMENT COMMAND SERIES ASSIGNMENTS

Command nos.[1]	Mode
500 - 550	SSO - RCS Model
551 - 574	RCS free flier 1, 24-jet system
575 - 599	RCS free flier 2, 24-jet system

[1]Command numbers 100 - 1100 are available

Figure 1-2 displays the organization of a simple two-vehicle simulation. The two vehicles involved are the Maneuverable Television (MTV) and the Space Shuttle Orbiter (SSO). The interaction between the two in the simulation displayed is solely through the "master clock" which represents the entire SOAP executive and scheduling system. The information written near the interface arrows describes some of the variables, in COMMON blocks, which are shared by the models. Any one of these variables is available for output at regular time intervals by specification to the SOAP system executive at the preprocessing stage. Such operations are transparent to the user once they are specified.

The simulation in figure 1-2 shows no sensors or passive vehicles, but these would form separate, additional blocks driven by the master clock. Pilots for the vehicles would form additional blocks within flight control. Information about each vehicle can be passed to any other through the COMMON blocks. Such passage would involve additional lines joining the flight control blocks of the vehicles. Vehicles can be added simply by replicating the blocks which model the vehicle features and by supplying the appropriate vehicle parameters.

Output from a SOAP simulation is typically initiated by the user by specifying those variables to be printed or plotted and the time interval at which they are to be recorded. The specifications are made during the preprocessing operation. Up to six dependent variables can be displayed as a function of a single "independent" variable within each plot. Three of the variables are represented by the position of symbols of the user's choosing, and three by the angles at which these symbols are drawn. Simplified plots can be produced on a line printer.

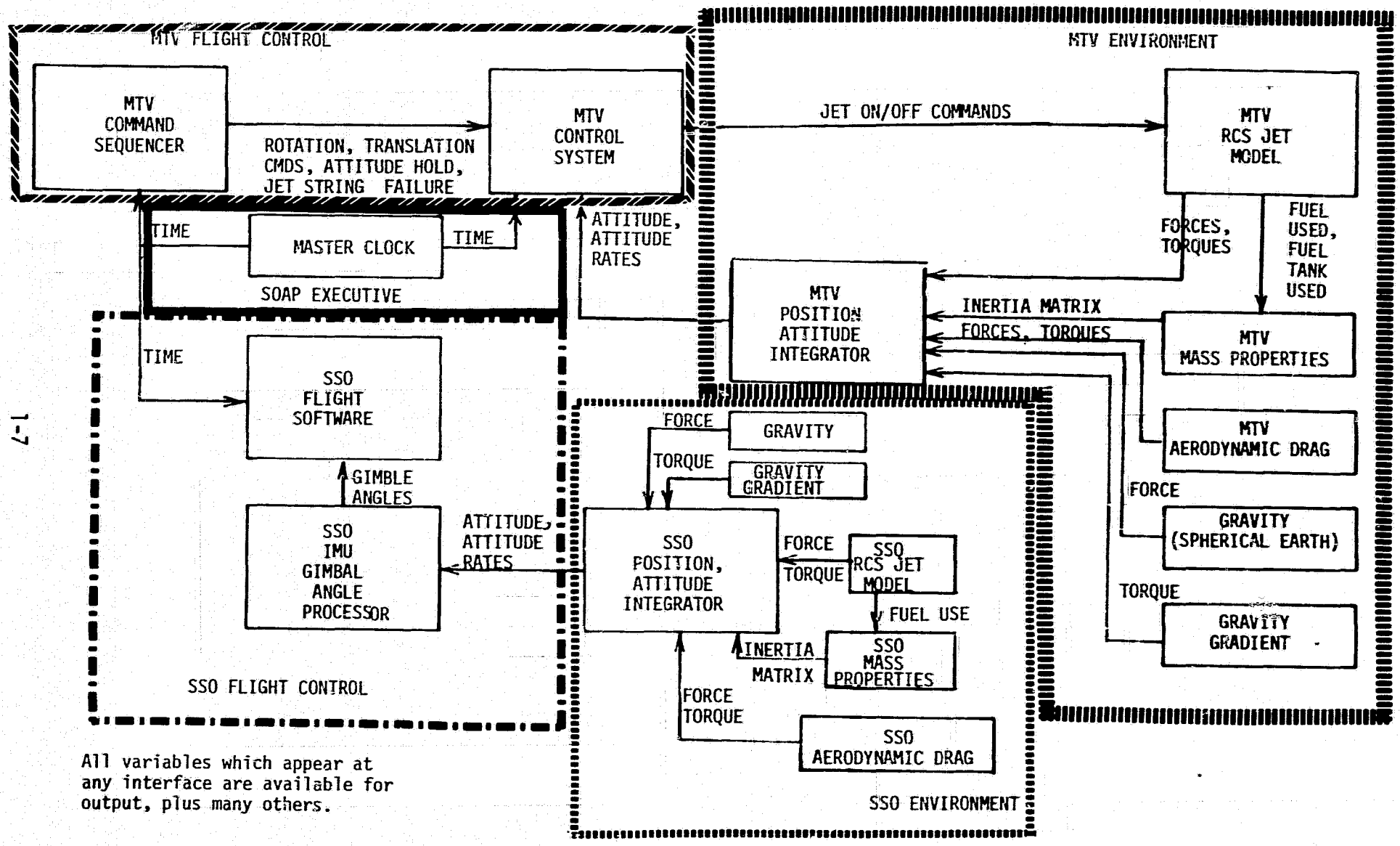


Figure 1-2.- SOAP organization, MTV and SSO.

The SOAP executive provides diagnostic capabilities in the form of a state image print and a restart capability. The state image print reproduces all of the COMMON block locations in decimal or Hollerith form. Such a print is typically supplied at the start and end of a simulation and can be requested (scheduled) at intervals during the simulation. The restart capability allows the user to initialize the simulation to replicate some point in the middle of another simulation; thus, a problem maneuver could be restarted immediately prior to a point of difficulty to avoid repeated simulation of an uninteresting time period. Instructions for obtaining a restarted run can be found in section 2.4.2.

Up to this point the SOAP description has been deliberately vague. The simulations which can be performed using the SOAP structure are not clearly limited. Those simulations which have already been performed and for which runstreams and models are already set up form a much more limited set. These simulations are documented by the runstreams in section 2 and by the models in section 4. It is hoped that these models will give the user a headstart in creating a model for any operations required.

Models of vehicle flight control systems are typically documented by flowcharts, scheduling algorithms, and program descriptions. The SSO software has, however, a special status. The operation of the SSO software is described in detail by Functional System Software Requirements (FSSR) documents. Much of the SSO software used by SOAP was developed for the SSFS and was designed to produce a functional model of the flight system. These routines, therefore, are not as extensively documented as are those routines developed for the SOAP alone.

The SOAP has roots in the SSFS. This document itself owes much of its development to documentation for the SSFS (refs. 1, 2, and 3). The authors would like to thank those who developed the SOAP framework and to refer the reader to this documentation as an example of a SOAP-type simulator.

2. SOAP USER'S GUIDE

2.1 PROGRAM FILES

This guide describes construction of runstreams which cause all four SOAP operations, i.e.,:
Preprocessing
Mapping
Simulation Execution
Postprocessing

These instructions are geared for the user of the UNIVAC 1110 with Exec 8.

The SOAP systems and library routines are maintained on tapes and secure files in the UNIVAC 1110. Both source and relocatable codes are available, though the casual user will interact with these routines only by scheduling and by mapping the relocatable elements. Typical selections of models and mapping instructions are available in section 2.2.

2.2 RUNSTREAM

Table 2-I displays the basic logical unit assignments for a SOAP run. Table 2-II describes the setup of the files required for a typical simulation. Typical lengths are associated with these files to ensure the availability of adequate storage.

TABLE 2-I.- INPUT/OUTPUT FILE REQUIREMENTS FOR SOAP EXECUTION

(a) Preprocessor

LU	Usage	Type	Purpose
3	Required	File	Output images for subroutine SELECT, CALMOD, and DUMMYS
4	Required	File	Output images for the collector directives
5	Required	Reader	Card input (can be file)
6	Required	Printer	Printed output (can be file)
29	Required	File	Scratch input/output

(b) Simulation

LU	Usage	Type	Purpose
3	Required	File	Scratch input/output
5	Required	Reader	Card input (can be file)
6	Required	Printer	Printed output (can be file)
7	Optional	Tape	State image dump input file; used for RESTART runs only
8	Optional	Tape	State image dump output file; used for RESTART or RECYCLE runs
11	Optional	File	Headlines print retention file; required if subroutine HDLNS is used
16	Optional	File	On-Orbit Astronaut File
20	Optional	File	Standard or primary plot dump unit; required if subroutine BLKPLT is used
29	Required	File	Scratch input/output
n	Optional	File	Secondary plot dump unit

TABLE 2-I.- CONCLUDED

(c) Plotter

LU	Usage	Type	Purpose
3	Required	File	Scratch input/output
5	Required	Reader	Card input
6	Required	Printer	Printed output
14	Optional	Tape	Plotter output tape required for CalComp plots
20	Optional	File	Standard or primary plot data input unit as furnished by the simulation step
29	Optional	File	Intermediate storage of plot data; required if the data is provided by tape and the PLOTER program is used
n	Optional	File	Secondary plot dump file, as furnished by the simulation step

TABLE 2-II.- SOAP RUNSTREAM SETUP

[When multiple entries with the same number appear, only one is used for a simulation]

```
1:      @RUN,R/TPR 170JG, E/3053, ED4-L44444,60,800
2:      @HDG,P PAGE HEADING INDENTIFYING RUN
3:      @ASG,AX ED4-L44444*SOAP800.
4:      @USE S., ED4-L44444*SOAP800.
5:      @ASG,AX ED-L44444*SOAPWORK.
6:      @USE SW., ED4-L44444*SOAPWORK.
7A:     @USE S5., ED4-L44444*SOAP805S1.
7B:     @USE S5., ED4-L44444*SOAP805D2.
8:      @ASG,AX S5.
9A:     @USE SB., ED4-L44444*SOAP2B0DS.
9B:     @USE SB., ED4-L44444*SOAP2B0DD.
9C:     @USE SB., ED-L44444*SOAP3B0DD.
10:     @ASG,AX SB.
11:     @ASG,T 3., F40/1/TRK/20
12:     @ASG,T 4., F40/1/TRK/20
13:     @ASG,T 8., F40/5/TRK/500
14:     @ASG,T 20., F40/20/TRK/500
15:     @ASG,T 29., F40/1/TRK/20
16:     @XQT S5. PREPRO
17:     @ADD,L SW.SCHED/SOAP
18A:    @ADD,L SB.SCHED/MTV
18B:    @ADD,L SB.SCHED/MTVD
18C:    @ADD,L SB.SCHED/MECD
18D:    @ADD,L SB.SCHED/MEC2
18E:    @ADD,L SB.SCHED/SSO
19:     @ADD,P 4.
20:     @ED,U SW.MAPIN, SW.MAPINN
21A:    ADD SB.MAP/MTV
21B:    ADD SB.MAP/MTVD
21:C    ADD SB.MAP/MECD
21D:    ADD SB.MAP/MEC2
```

TABLE 2-II.- CONCLUDED

```
21E:    ADD SB. MAP/SSO
22:     @PREP SW.
23:     @MAP,S SW.MAPINN,TPF$.SOAP
24:     @XQT SOAP
25:     (SIMULATION INPUT DATA CARDS - see section 2.4)
26:     @JSC*CALLUP.TAPELABEL
27:     @ASG,TJ $P-P$TAPE,U,,21_._FRM02/31 PLT DESC.
28:     @USE 14., $P-P$TAPE.
29:     @REWIND 14.
30:     @XQT S5.PLOTER
31:     (PLOTTER DATA CARDS - see section 2.5)
32:     @FIN
```

A description of each of the cards shown in table 2-II follows. A more detailed description of the control statements and their options can be found in UNIVAC system documentation. In the following descriptions, each "card" represents one line of instructions.

Card 1:

Image: @RUN,P/O RunID,AcctID,ProjID,RT,PG/CD NAME N

Purpose: To initiate the run and provide identification and accounting information to the system.

Format:

@RUN - Required mnemonics for the run card.

P - Priority code for the run. It consists of a single letter: R, S, U, V, or Z. R has the highest priority; Z is the system default value.

O - Run card options.

RunID - Run identification field, which must consist of the box number, two of the user's initials, and a letter designator as required for uniqueness; e.g., 170NBB. Six characters is the maximum number.

AcctID - Account identification field, which is composed of the first letter of the user's organization code and the user's project number separated by a slash; e.g., E/3096.

ProjID - Project identification field, which consists of the user's organization code and employee number separated by a minus-sign; e.g., EH2-L55555.

RT - Maximum run time in minutes.

PG/CD - Maximum number of pages (PG) and cards (CD) output for this run. If no cards are expected, the /CD should be omitted.

NAME - First six characters of the user's last name. The field begins in column 61. (optional)

N - A 6 in column 80, designating UNIVAC 1110-6 EXEC 8 processing.

Card 2:

Image: @HDG,P MESSAGE

Purpose: To provide a heading message to be printed at the top of each page of printer output along with a page number and date.

Format:

@HDG - Required mnemonics for the heading control statement.

P - Heading statement option specifying that the page count will begin with 1.

MESSAGE - Heading that is to appear at the top of each page. The heading is limited to 96 characters, including blanks. NOTE: A period (.) cannot be used in the message because the portion of the message following the period may not be printed.

Card 3:

Image: @ASG,A ED4-L44444*SOAP800.

Purpose: To assign the SOAP System to this run.

Format:

@ASG - Required mnemonics for the assign card.

A - Assign card option: The A indicates that the assign is for an existing cataloged file.

ED4-L44444* SOAP800. - Complete file name of the cataloged file containing the SOAP System.

Card 4:

Image: @USE S., ED4-L44444*SOAP800.

Purpose: To link the internal file name S. to the external file name ED4-L44444*SOAP800. The MAPIN program references file S. as the standard program file for the SOAP system.

Format: @USE - Required mnemonics for the use control statement.

S. - Internal file name.

ED4-... - External file name.

Card 5:

Image: @ASG,A ED4-L44444*SOAPWORK

Purpose: To assign the file containing flight software, Environment, and utility routines used for most simulations.

Format: @ASG - Required mnemonics for the assign card.

A - Assign card option indicating an existing cataloged file.

X - Indicates exclusive use of this file by this run.

ED4-L44444* SOAPWORK - Complete file name

Card 6:

Image: @USE W., ED4-L44444*SOAPWORK.

Purpose: To link the internal file name W. to the external file name ED4-L44444*SOAPWORK. The MAPIN program refers to file W. as the standard program file.

Format: @USE - Required mnemonics for the use control statement.

W. - Internal file name.

ED4-... - External file name.

Card 7:

Image: @USE,S5,ED4-L44444*SOAP805S1.
@USE,S5,ED4-L44444*SOAP805D2.

Purpose: To link the internal file name S5. to the external file name ED-L44444*SOAP805S1 or ED4-L44444*SOAP805D2. SOAP805S1 is the single precision, two-vehicle program file containing the pre-processor element. SOAP805D2 is the double precision, multi-vehicle (up to 1 SSO and 2 MTVs) program file version file version of SOAP805S1.

Format: @USE - Required mnemonics for the use control statement.

S5. - Internal file name.

ED4-L44444*SOAP805S1.
or } external file name.
ED4-L44444*SOAP805D2.

Card 8:

Image: $ASG,A S5.

Purpose: To assign the file containing the PREPRO and PLOTER elements and various input/output routines.

Format: @ASG - Required mnemonics for the assign card.

A - Assign card option indicating an existing cataloged file.

S5. - Internal file name attached to either ED4-L44444*SOAP805S1 or ED4-L44444*SOAP805D2.

Card 9:

Image: @USE SB., ED4-L44444*SOAP2B0DS.

@USE SB., ED4-L44444*SOAP2B0DD.

@USE SB., ED4-L44444*SOAP3B0DD.

Purpose: To link the internal filename SB. to one of three external file name as follows:

A) ED4-L44444*SOAP2BODS - Program file containing environment and flight software routines for a single precision two-vehicle simulation.

B) ED4-L44444*SOAP2BODD - Same as A) only in double precision.

C) ED4-L44444*SOAP3BODD - Same as B) only for 3 vehicles instead of 2.

Format: @USE - Required mnemonics for the use control statement.

SB. - Internal file name

ED4-L44444*SOAP2BODS
ED4-L44444*SOAP2BODD
ED4-L44444*SOAP2BODD
} external file, use one

Card 10:

Image: @ASG, A SB.

Purpose: To assign the file containing the appropriate flight software and environment routines for the simulation.

Format: @ASG - Required mnemonics for the assign card.

A - Assign card option indicating and existing cataloged file.

SB. - Internal file name attached to one of the following:

ED4-L44444*SOAP2BODS.

ED4-L44444*SOAP2BODD.

ED4-L44444*SOAP3BODD.

Card 11:

Image: @ASG,T 3.,F40/1/TRK/10

Purpose: To assign a FASTRAND-formatted mass storage file to this run. The file is used for Preprocessor output.

Format:

@ASG	- Required mnemonics for the assign card.
T	- Assign card option denoting a temporary file assignment.
3.	- Internal file name assigned to the file.
F40	- Code specifying that this assign statement is for a FASTRAND-formatted mass storage file and identifying the specific device required.
1	- Integer specifying the minimum amount of mass storage required for this file.
TRK	- Increment or granule size for this assignment. One track (TRK) has 1792 words.
10	- Integer specifying the maximum amount of mass storage required for this file (10 tracks = 17,920 words).

Card 12:

Image: @ASG,T 4.,F40/1/TRK/10

Purpose: To assign a FASTRAND-formatted mass storage file to this run. The file is used for Preprocessor output.

Format:

@ASG	- Required mnemonics for the assign card.
T	- Assign card option denoting a temporary file assignment.
4.	- Internal file name assigned to the file.
F40	- Code specifying that this assign statement is for a FASTRAND-formatted mass storage file and identifying the specific device required.
1	- Integer specifying the minimum amount of mass storage required for this file.
TRK	- Increment or granule size for this assignment. One track (TRK) has 1792 words.
10	- Integer specifying the maximum amount of mass storage required for this file (10 tracks = 17,920 words).

Card 13:

Image: @ASG,T 8.,F40/5/TRK/500

Purpose: To assign a FASTRAND-formatted mass storage file to this run. The file is used for state image dumps for recycle and/or restart purposes.

Format:

@ASG	- Required mnemonics for the assign card.
T	- Assign card option denoting a temporary file assignment.
8.	- Internal file name assigned to the file.
F40	- Code specifying that this assign statement is for a FASTRAND-formatted mass storage file and identifying the specific device required.
5	- Integer specifying the minimum amount of mass storage required for this file.
TRK	- Increment or granule size for this assignment.
500	- Integer specifying the maximum amount of mass storage required for this file.

Card 14:

Image: @ASG,T 20.,F40/20/TRK/200

Purpose: To assign a FASTRAND-formatted mass storage file to this run. The file is used for plot dump output.

Format:

@ASG	- Required mnemonics for the assign card.
T	- Assign card option denoting a temporary file assignment.
20.	- Internal name assigned to the file.
F40	- Code specifying that this assign statement is for a FASTRAND-formatted mass storage file and identifying the specific device required.
20	- Integer specifying the minimum amount of mass storage required for this file.

TRK - Increment or granule size for this assignment.

200 - Integer specifying the maximum amount of mass storage required for this file.

Card 15:

Image: @ASG,T 29.,F40/1/TRK/10

Purpose: To assign a FASTRAND-formatted mass storage file to this run. The file is used for scratch input and output.

Format:

@ASG - Required mnemonics for the assign card.

T - Assign card option denoting a temporary file assignment.

29. - Internal name assigned to the file.

F40 - Code specifying that this assign statement is for a FASTRAND-formatted mass storage file and identifying the specific device required.

1 - Integer specifying the minimum amount of mass storage required for this file.

TRK - Increment or granule size for this assignment.

10 - Integer specifying the maximum amount of mass storage required for this file.

Card 16:

Image: @XQT S5.PREPRO

Purpose: To execute the SOAP Preprocessor program.

Format:

@XQT - Required mnemonics for the execute control statement.

S5. - Name of the file containing the absolute element PREPRO.

PREPRO - Name of the absolute element to be executed.

Card 17, 18 (A, B, C, D, E): These cards represent the data images for the Preprocessor program. The specific contents of this deck are discussed in section 2.3.

Card 19:

Image: @ADD,P 4.

Purpose: To add the images on logical unit 4 to the runstream at this point.

Format:

@ADD	- Required mnemonics for the add control statement.
P	- Add statement option indicating that the ADD statement appears in the listing.
4.	- External file name of the file to be added to the runstream.

See section 2.3.1 for the options affecting the output listing of routine SELECT and the Memory Allocation Processor (MAP) for the SOAP.

Card 20:

Image: @ED,U SW.MAPIN, SW.MAPINN

Purpose: To call the edit processor to update the existing element MAPIN and to name the new updated element MAPINN. MAPINN will contain a set of collector directive cards as specified by card 21.

Format:

@ED	- Required mnemonics to call edit processor.
U	- Edit card option indicating an update of the element.
SW.	- Internal file containing the element MAPIN.
MAPIN	- Element containing some collector directive cards standard to all runs.
MAPINN	- Updated element containg all necessary collector directive cards to produce the absolute element SOAP.

Card 21:

Image: ADD SB.MAP/MTV

ADD SB.MAP/MTVD

ADD SB.MAP/MECD

ADD SB. MAP/MEC2

ADD SB.MAP/SSO

Purpose: To add to the existing set of collector directives in element MAPIN an additional set of directives corresponding to the vehicle(s) participating in the simulation.

Format: ADD - Edit command that adds all lines of the element specified to the element being updated.

SB. - Internal file name containing the map element.

MAP/VEHICLE - Element name and version of lines to be added to the updated element.

MTV
MTVD
MECD
MEC2
SSO

} substitute for vehicle, no parentheses

Card 22:

Image: @PREP SW.

Purpose: To create an entry point table for file SW.

Format: @PREP - Required mnemonics to call FURPUR processor.

SW. - Internal file name.

Card 23:

Image: @MAP,S SW.MAPINN,TPF$.SOAP

Purpose: To use the set of collector directives in element MAPINN and create an absolute executable element, SOAP, and put it in file TPF$.

Format: @MAP - Required mnemonics to call the MAP processor.

S - MAP processor option requesting a short listing of storage allocation and octal addresses of the relocatables included.

SW. - Internal file containing MAPINN.

MAPINN - Set of collector directive instructions.

TPF$ - Temporary file to contain the absolute element SOAP.

SOAP - The absolute element to be executed.

Card 24:

Image: @XQT SOAP

Purpose: To execute the SOAP.

Format: @XQT - Required mnemonics for the execute control statement.

SOAP - Name of the absolute element to be executed.

Card 25: This card represents the input data set for the SOAP. The composition of this data set is discussed in section 2.4.3.

Card 26:

Image: @JSC*CALLUP.TAPELABEL

Purpose: To automatically generate gummed labels and the corresponding tape library transaction for new save tapes. This card must immediately precede the tape assign card.

Format: @JSC*CALLUP. - Complete file name of the CALLUP processor file.

TAPELABEL - Processor for tape labeling and saving.

Card 27:

Image: @ASG,T $P-P$TAPE.,U,,21 . FRM02/31PLT label.

Purpose: To create a CalComp plot label and to create a save plot tape.

Format: @ASG - Required mnemonics for the assign card.

T - Assign card option denoting a temporary file assignment.

$P- - The required first three characters of the filename portion of the file specified if the tape is to be saved is an "X" bin tape requiring peripheral processing.

P$TAPE. - Tape file name. If D option is used, this file name must be used.

U	- Code specifying that this assign statement is for a magnetic tape device and identifying the specific type of unit required.
21	- Retention period in days.
FRM02/31PLT	- Description field, which must limit entire field to 24 characters and contain type of form to be used for the plot (02), number of plots, label for tape.

Card 28:

Image: @USE 14., $P-P$TAPE.

Purpose: To link the internal file name (14.) to the external file name ($P-P$TAPE) so that FORTRAN references to logical unit 14 are equivalent to system references to PLOT.

Format:

@USE	- Required mnemonics for the use control statement.
14.	- Internal or FORTRAN file name.
$P-P$TAPE.	- External or system file name.

Card 29:

Image: @REWIND 14.

Purpose: To rewind the tape file PLOT. to the load point.

Format:

@REWIND	- Required mnemonics for the rewind control statement.
14.	- Internal file name of the tape file to be rewound.

Card 30:

Image: @XQT S5.PLOTER

Purpose: To execute the SSFS plotter program.

Format:

@XQT	- Required mnemonics for the execute control statement.
S5.	- Name of the file that contains the absolute element PLOTER.
PLOTER	- Name of the absolute element to be executed.

Card 31: This card represents the data set for the SSFS plotter program. The composition of this data set is discussed in section 2.5.

Card 32:

Image: @FIN

Purpose: To indicate the end of the runstream.

Format: @FIN - Required mnemonics for the finish control statement.

2.3 PREPROCESSOR DATA

Input to the Preprocessor consists of six data sets: a SELECT/MAP specification card, Environment configuration specification cards, flight software configuration specification cards, subroutine scheduling cards, print/plot output definition cards, and subroutine MAXMIN output definition cards.

On all Preprocessor data cards the capability exists to add a comment to each card if the comment is preceded by the $ character. For this reason, care should be taken not to use TPF$ as a file name to load from. A $ character placed in card column 1 designates the whole card as a comment.

The Preprocessor uses three mass storage work files: 3, 4, and 29. Logical unit 29 is the scratch file used to decode and build card images. Logical units 3 and 4 contain output pertinent to the main simulation. On logical unit 3, the Preprocessor writes the following card images:

- @ED,IN CALMOD
- (Subroutine CALMOD code)
- @ED,IN DUMMYS
- (Subroutine DUMMYS code)
- @FOR,N CALMOD
- @FOR,N DUMMYS
- @ED,IN SELECT
- (Subroutine SELECT code)
- @FOR,N SELECT

2.3.1 SELECT/MAP

This data card must be inserted only as the first card of the preprocessor data deck to affect the printing of the subroutines SELECT CALMOD, and DUMMYS, and the MAP of the SOAP. The format of this card begins in column 1 and is SELECT(N),MAP(N)

The control characters "N, S, L, X" are placed in parentheses as user specifications. An "N" specification will cause the most abbreviated listing to be generated. The "N" specification is the default for SELECT. An "S" specification will cause a moderately comprehensive listing. An "L" specification will cause the most comprehensive listing to be generated. The "X" specification applies only to the MAP, and must be specified so that no MAP will be generated. Following this card, the Environment configuration specification data set must be included to ensure the proper generation of routines CALMOD and DUMMY. This data is stored in their respective MAP/VEHICLE elements in the SR. file.

2.3.2 ENVIRONMENT CONFIGURATION

The Environment configuration specification data set (see table 2-III) is used to define the Environment model(s) and the file(s) where the model(s) reside. If no models are specified, the Preprocessor will generate the CALMOD subroutine which contains a call to the active vehicle (ACTVEH) driver, and the DUMMYS subroutine which contains entry points for all other models. In all cases where a dummy ACTVEH is requested, a dummy version of subroutine ACTVEH will be loaded from the SOAP System library.

In the case where an Environment model is to be included in an overlay segment other than the "ROOT" segment, an Environment specification card must be included for that model and the version of the model requested must be specified as "OVRLAY." In the particular segment where the model is to be loaded, the model will be specified with the model's real version name. Also, an additional specification can be made to request DCOML, the routine used to define the length of all the COMMON blocks to be loaded from a file other than "W."

Each image is free-field and only one specification should be made per card.

A comment can be added to a card if the comment is preceded by a $ character. A $ character in column 1 designates the whole card as a comment.

All models desired must be specified for a simulation configuration. A dummy entry point will be generated for all models not specifically requested.

TABLE 2-III.- ENVIRONMENT CONFIGURATION SPECIFICATION DATA SETS

Card image	Action
SELECT(S),MAP(X)	Generate a moderately comprehensive listing of CALMOD, DUMMYS, and SELECT. Do not generate MAP directives.
ACTVEH=MYFILE.AVEHIX $ USE MY AV	Load active vehicle model version AVEHIX from file MYFILE.
RCS=W.RCS22	Load Reaction Control System model version RCS22 from file W.
GRAV=.GRAV2	Load gravity model version GRAV4 from default file W.
AERO=AERO37	Load aerodynamic forces model version AERO37 from default file W.
.	Other Environment models can be specified here. Any models not specified will have dummy routines supplied by the Preprocessor.
.	
.	
GGTORK=OVRLAY $ GGT IN SEG 2	Inform the Preprocessor that the gravity gradient torque model will be loaded in a segment, and to generate a call to GGTORK in CALMOD rather than an entry point in DUMMYS.
DCOML=W.EDCOML	DCOML is specified here to ensure loading at the beginning of the MAP. If not specified, version DCOML will be loaded from W.
*EOF	Required terminator for data set.

All names are limited to six characters each.

The minimum input for this data set is an *EOF card.

The general form of the model selection equation is as follows:

Image: $model_1$=$file_1$.$version_1$

Purpose: To define the Environment configuration specification for a particular mission phase.

Format: Any number of free-field images consisting of a model selection equation. The order of the equations is irrelevant.

$model_i$ - Field used to identify the type of model that is being specified by this equation. (This mnemonic is defined in table 2-IV.)

$file_i$. - Optional field which, if used, allows the user to specify the file where this model resides. "$file_i$" is limited to six characters. If the field is omitted, file W is assumed. If only the period (.) is used, the default file is W.
CAUTION: do not use the name TPF$.

$version_i$ - Version name of the model to be used in this simulation. Dummy models may be selected by specifying the version to be DUMMY or version number 00. For a dummy specification, an entry point is generated in subroutine DUMMYS. Available models are described in this document.

The format of the *EOF card is:

Image: *EOF

Purpose: To denote the end of a data set within a data stream.

Format: *EOF - Required mnemonic for the end of a data set.

TABLE 2-IV.- ENVIRONMENT MODEL MNEMONICS AND DEFINITIONS

MODEL * ($model_i$)	VERSION ($version_i$)	MODEL DEFINITION
ACTVEH	AVEH	Active Vehicle
IMU	IMU	Inertial Measurement Unit
RCS	RCS	Reaction Control System
TVC	TVC	Thrust Vector Control
THRUST	THR	Engine Thrust
ACS	ACS	Aerodynamic Control System
ACCEL	ACC	Accelerometer
SNSCTL	SNSC	Sensor Control
MASPRO	MASS	Mass Properties
AERO	AERO	Aerodynamic Forces
GRAV	GRAV	Gravity
ATMOS	ATM	Atmosphere
SLOSH	SLSH	Fuel Slosh
BEND	BEND	Vehicle Bending
GYRO	GYRO	Gyroscopic Sensors
PASVEH	PVEH	Passive Vehicle
GUST	GUST	Wind Turbulance
HORIZS	HORS	Horizon Sensor
STARTR	STAR	Star Tracker
RENDES	RSEN	Rendezvous Sensor
ORBALT	OALT	Orbital Altimeter
NAVAID	NAVA	Navigation Aids
GRADAR	GRAD	Ground Radar
RADALT	RALT	Radar Altimeter
GGTORK	GGT	Gravity Gradient Torque
ORBPAR	ORBP	Orbital Parameters
AIRDAT	AIRD	Air Data Sensor
WIND	WIND	Wind Effects
RNGNAV	RNGN	Range Navigation
LNDAID	LNDA	Landing Aids
BARALT	BALT	Barometric Altimeter
LNDGR	LNDG	Landing Gear

*Not all models available except as dummies.

2.3.3 SCHEDULING

This Preprocessor data subset specifies the time-dependent flight control subroutines and print/plot routines to be executed and defines their execution characteristics, such as frequency, priority, and start and stop times.

As in the previous data sets, a comment can be added to a card if the comment is preceded by a $ character. A $ character in column 1 designates the whole card as a comment. Unlike the other data sets, all the comments associated with this data set are printed before the scheduling information table is printed.

Each image is free-field and unique in that each image is processed during execution as a single request. Therefore, multiple scheduling intervals for one routine can overlap.

All names are limited to six characters each.

The data image is defined as follows:

Image: NAME,PRIOR,DELTAT,STARTT,STOPT,NAMEB,NAMEA,TIME1,TIME2

Purpose: To define the execution characteristics of the time-dependent flight software and print/plot routines.

Format: Free-field images with comments.

- NAME - Name of the flight control subroutine, entry point, print/plot routine, or other schedulable routine.
- PRIOR - Relative priority for the execution of subroutine NAME. A value of 0.0 represents the highest priority. If two or more routines are scheduled for execution at the same time, the one having the highest priority will be executed first.
- DELTAT - Time interval, in seconds, between executions of routine NAME. If the field is left blank or input as zero, the value of DELTAT will be set to 99999.999.

STARTT - Simulation time, in seconds relative to simulator initialization, at which the flight software routine (NAME) is to start execution. A value of -1.0 in this field indicates that the routine will be scheduled by another routine during the simulation.

STOPT - Time, in seconds, at which NAME is to cease executing. If the field is blank or zero, STOPT will be set to $1x10^{10}$.

NAMEB - Name of the subroutine that is to be executed just prior to the execution of NAME. This field is optional.

NAMEA - Name of the subroutine that is to be executed immediately after the execution of NAME. The field is optional.

TIME1 - Start time for NAMEA and/or NAMEB. If omitted, the start time will be STARTT. If TIME1 is specified, TIME2 must also be specified.

TIME2 - Stop time for NAMEA and/or NAMEB. If TIME1 and TIME2 are omitted, the stop time will be STOPT.

Like the other data sets, it must be terminated by an end-of-file card.

Image: *EOF

Purpose: To denote the end of a data set within a data stream.

Format: *EOF - Required mnemonics for the end-of-data set statement.

To ensure that the calling linkage to a routine or entry point is generated, a schedule data card must be present for each routine or entry point that has been scheduled or is to be scheduled for execution on the SOAP. A start time of -1.0 indicates that the routine will be scheduled by some flight control algorithm during the simulation. More than one schedule data card may be furnished for the same routine as long as each has a start time greater than or equal to 0.0. Minimum input for this subset is a schedule data card for subroutine TERMIN (the termination routine for SOAP) and the end-of-file card. The TERMIN card is required to prevent a "run-away" simulation. The Preprocessor checks for the presence of the TERMIN card and inhibits the simulation if it is not found.

When an initial simulation time bias is given on the simulation input card INIT, all scheduled start and stop times will be altered by the initial time value.

A restriction which must be observed is that schedule data cards for Preprocessor-written routines cannot specify whether other routines are to be run as "before" or "after" routines. Preprocessor-written routines include print, plot, and MAXMIN requests which are entry points in SELECT. However, the user can include these routines as "before" or "after" routines on schedule data cards of non-Preprocessor written routines. Routines scheduled as "before" or "after" routines should not exit by calling FREQEJ. If they do, each time one of these routines is called, the primary routines will be rescheduled. Samples of routine scheduling cards are shown in table 2-V.

Default values are assigned for the various times and priority if they are not specified. The defaults are:

PRIORITY	default is	0.0
DELTA TIME	default is	99999.999
START TIME	default is	0.0
STOP TIME	default is	1.0E+10

an exception is made for routine SIDUMP (state image dump). Whatever the scheduled or default priority associated with scheduling SIDUMP, it is changed to 1.0E+38 when SIDUMP is actually scheduled on the waitlist.

Another SOAP system routine that needs special attention is SIPRIN (state image print). Like SIDUMP, SIPRIN can be scheduled or called. SIPRIN when executed will print all the SOAP standard COMMON blocks listed in table 2-VI. Entry points that can be scheduled or called are also listed in table 2-VI. All of these entry points are in SIPRIN.

Note that one subroutine is scheduled automatically for each simulation. Subroutine KALNOW sets the immediate Environment update flag, LECCOM(740), to force a call to the Environment.

TABLE 2-V.- SAMPLE SCHEDULE DATA CARDS

Card	Card image	Action
1	$ COMPLETE CARD IMAGE COMMENT	A $ character in card column 1 results in the whole card treated as a comment.
2	ETGUID	A routine name specification only results in a priority of 0.0, a delta time of 99999.999, a start time of 0.0, and a stop time of 1.0E+10.
3	PLOTR,1.0,0.40	Routine PLOTR will execute every 0.40 second with a priority of 1.0. The start time is 0.0 and the stop time is 1.0E+10.
4	TAGOFT,3.0,0.96,-1.0 $ TAEM GUIDANCE	Routine TAGOFT will be started at some time during the run as designated by the -1.0 start time, and will execute every 0.96 second with a priority of 3.0.
5	QDREFC,8.95,0.96,0.0,1200.0	Routine QDREFC will execute every 0.96 second with priority 8.95 beginning at time 0.0 and stopping at time 1200.0.
6	SIDUMP,,96.0	A state image dump will be made every 96.0 second with a priority of 1.0E+38 beginning at time 0.0 and stopping at 1.0E+10.
7	TERMIN,100.0,,100.0,,LASTP2	Routine TERMIN will be executed every 99999.999 seconds with a priority of 100.0 beginning at time 100.0. Once TERMIN is called, the run is terminated. However, routine LASTP2 will be called immediately before the call to TERMIN.
8	CALCUL,1.0,2.0,10.0,100.0,P1,P2,50.0,60.0	Routine CALCUL will be executed every 2.0 seconds with a priority of 1.0 starting at time 10.0 and stopping at time 100.0. Routines P1 and P2 will be executed immediately before and after CALCUL, respectively, starting at time 50.0 and stopping at time 60.0. If the last two times were not specified, then P1 and P2 would be executed every time CALCUL is executed.
9	*EOF	Required data set terminator.

TABLE 2-VI.- SOAP COMMON BLOCKS

COMMON block	Definition	S/D	Entry points in SIPRIN
LECCOM	Executive COMMON	S	LECCOP
FSCOM	Flight Control (FC) COMMON	S	FSCOMP
ENVCOM	Environment Executive COMMON	S	ENVCOP
ACTVEC	ACTVEH COMMON	S,D	ACTVEP
IMUC	IMU COMMON	S,D	IMUP
RCSC	RCS COMMON	S,D	RCSP
TVCC	TVC COMMON	S,D	TVCP
THRUSC	THRUST COMMON	S,D	THRUSP
ACSC	ACS COMMON	S,D	ACSP
ACCELC	ACCEL COMMON	S,D	ACCELP
SNSCTC	SCSCTL COMMMON	S,D	SNSCTP
MASPRC	MASPRO COMMON	S,D	MASPRP
AEROC	AERO COMMON	S,D	AEROP
GRAVC	GRAV COMMON	S,D	GRAVP
ATMOSC	ATMOS COMMON	S,D	ATMOSP
SLOSHC	SLOSH COMMON	S,D	SLOSHP
BENDC	BEND COMMON	S,D	BENDP
GYROC	GYRO COMMON	S,D	GYROP
PASVEC	PASVEH COMMON	S,D	PASVEP
GUSTC	GUST COMMON	S,D	GUSTP
HORIZC	HORIZS COMMON	S,D	HORIZP
STARTC	STARTR COMMON	S,D	STARTP
RENDEC	RENDES COMMON	S,D	RENDEP
ORBALC	ORBALT COMMON	S,D	ORBALP
NAVAIC	NAVAID COMMON	S,D	NAVAIP
GRADAC	GRADAR COMMON	S,D	GRADAP
RADALC	RADALT COMMON	S,D	RADALP
GGTORC	GGTORK COMMON	S,D	GGTORP
ORBPAC	ORBPAR COMMON	S,D	ORBPAP
AIRDAC	AIRDAT COMMON	S,D	AIRDAP

S = Single precision
D = Double precision

TABLE 2-VI.-CONCLUDED

COMMON block	Definition	S/D	Entry points in SIPRIN
WINDC	WIND COMMON	S,D	WINDP
RNGNAC	RNGNAV COMMON	S,D	RNGNAP
LNDAIC	LNDAID COMMON	S,D	LNDAIP
BARALC	BARALT COMMON	S,D	BARALP
FSC1	Additional FC COMMON	S	FCS1P
FSC2	Additional FC COMMON	S	FSC2P
FSC3	Additional FC COMMON	S	FSC3P
FSC4	Additional FC COMMON	S	FSC4P
FSC5	Additional FC COMMON	S	FSC5P
LNDGRC	Landing Gear COMMON	S,D	DATBUP
DATBUS	Environment-to-flight control Bus COMMON	S,D	DPDBCP
DPDBC	Environment-to-flight control Bus COMMON	D	DPFSCP
DPFSC	FC COMMON	D	DFSC1P
DPFSC1	Additional FC COMMON	D	DAVEHP
DAVEHC	ACTVEH COMMON	D	DIMUP
DIMUC	IMU COMMON	D	DMASSP
DMASSC	MASPRO COMMON	D	DGRAVP
DGRAVC	GRAV COMMON	D	FS1CP
FS1C	VEH#1 FC COMMON	S,D	FS2CP
FS2C	VEH#2 FC COMMON	D	RCS1CP
RCS1C	VEH#1 RCS COMMON	S,D	RCS2CP
RCS2C	VEH#2 RCS COMMON	D	ACTV1P
ACTV1C	VEH#1 ACTVEH COMMON	S,D	ACTV2P
ACTV2C	VEH#2 ACTVEH COMMON	D	MASP1P
MASP1C	VEH#1 MASPRO COMMON	S,D	MASP2P
MASP2C	VEH#2 MASPRO COMMON	D	HORZ1P
HORZ1C	VEH#1 HORIZ COMMON	D	HORZ2P
HORZ2C	VEH#2 HORIZ COMMON	D	AER1CP
AER1C	VEH#1 AERO COMMON	S,D	AER2CP
AER2C	VEH#2 AERO COMMON	S	

If there are no print/plot routine requests, the *EOF terminating the scheduling data set should be followed immediately by a card image with "QUIT" beginning in column 1.

2.3.4 PRINT/PLOT

This data subset allows the user to build print and/or plot dump subprograms which may either be scheduled or called. The user may specify up to 90 print routines and up to 10 plot dump routines. Any unique name (up to six characters) may be used for the name of the print routine. Plot dump routine names are restricted to one of the following: PLOTR, PLOTR1, ..., PLOTR9. Variables to be printed and/or plotted must be selected from any location of any labeled COMMON block. The user specifies mnemonics (up to six characters) for each parameter to be printed or plotted. Note that no plotting actually takes place during the simulation, but that variables to be plotted are written on an output device (tape or drum) to be used by one of the SOAP postprocessors.

Print or plot dump routine data consists of two parts: a card defining the name of the routine and a card or cards indicating the variables to be printed or plotted. Each print or plot dump routine set must be terminated with an end of file (*EOF). The format of the card images follows.

Image: NAME

Purpose: To define the name of the print or plot dump routine which will be built by the Preprocessor. The routine may be scheduled and/or called.

Format: Fixed-field (A6) format. NAME must be left-justified in the field; i.e., it must start in column 1.

NAME - Name of the routine. A print routine may be named with any unique mnemonics (up to six characters). Plot routine names must be one of the following: PLOTR, PLOTR1, ..., PLOTR9. If NAME is the word QUIT, it denotes the end of the Preprocessor output routine requests and the end of the Preprocessor input data set.

Image: VARNAM = COMMON(I) OP $\pm$ SF,FMT $ COMMENT

Purpose: To define the parameters to be printed or plotted.

Format: Free-field format.

VARNAM - Mnemonics (up to six characters) used to identify the parameter in the printed output or used as plotter input.

COMMON - Name of the labeled COMMON block in which the parameter is located.

I - Location in the COMMON block where the output parameter is stored. This code is limited to the range from 1 to 9999.

OP - Optional operation code used to scale the value in COMMON(I). OP is one of the four arithmetic operators: +,-,*, or /.

$\pm$ - Optional sign + or - can be assigned to the scale factor.

SF - Optional field that allows the user to input a scale factor which will be applied to the data before it is output. Only the first 18 characters of SF are used. SF must be the same type as the data. In addition, 19 mnemonics are available for SF, providing primarily English-metric conversion factors. The mnemonics and corresponding conversion factors are given in table 2-VII. Also, the scale factor can be another COMMON name and location. In this case, no effort is made to decode the name and location. Therefore, the name must appear at least once in the "COMMON(I)" position in the image above. This appearance can be in any of the print/plot images specified. When using the COMMON blocks LECCOM, IMUC, MASPRC, NAVAIC, LNDAIC, and LNDGRC as a scale factor, the first letter must be replaced with a Q. This is true only when used as a scale factor.

,FMT - Optional field that defines the format of the output parameter. If FMT is specified, the comma (,) is required. FMT may be E or F for floating point, I for integer, or B for octal. If FMT is omitted, the standard FORTRAN convention

TABLE 2-VII.- SCALE FACTOR MNEMONICS AND RESULTING VALUES

TO CONVERT FROM	USE SF MNEMONICS	LOCATION IN CONST COMMON	CONVERSION FACTOR
Inches to meters	ITM	29	* 0.0254 m/in.
Degrees to radians	DTR	31	0.01745329252 rad/deg
Radians to degrees	RTD	32	57.29577951 deg/rad
Feet to meters	FTM	33	* 0.3048 m/ft
Meters to feet	MTF	34	3.280839895 ft/m
Pounds to kilograms	PTK	35	* 0.453592370 kg/lb
Kilograms to pounds	KTP	36	2.204622622 lb/kg
Pounds (force) to newtons	PTN	37	4.448221615 N/lbf
Newtons to pounds (force)	NTP	38	0.2248089431 lbf/N
Slugs to kilograms	STK	39	14.59390294 kg/slug
Kilograms to slugs	KTS	40	0.06852176586 slugs/kg
Foot-pounds (force) to newton-meters	FTN	41	1.355817948 N-m/ft-lbf
Newton-meters to foot-pounds (force)	NTF	42	0.7375621493 ft-lbf/N-m
Pounds to slugs	PTS	43	0.03108095017 slugs/lb
Slugs to pounds	STP	44	32.17404856 lb/slug
Kilogram-meters^2 to slug-feet^2	IME	45	0.7375621493 slug-ft^2/kg-m^2
Slug-feet^2 to kilogram-meters^2	IEM	46	1.355817948 kg-m^2/slug-ft^2
Newtons/meter^2 to pounds (force)/foot^2	PME	47	0.02088543423 (lbf/ft^2)/(N/m^2)
Pounds (force)/foot^2 to newtons/meter^2	PEM	48	47.88025898 (N/m^2)/(lbf/ft^2)

*Indicates exact number.

is used; that is, floating-point data is assumed for all variables unless they begin with I, J, K, L, M, or N, in which case the integer format is assumed.

$ - Optional field that terminates the data scan on the card. The remainder of the card may be used for comments. The $ can be incorporated into card column 1 designating the whole card as a comment. The $ in column 1 must not be used before the NAME card.

COMMENT - Optional comments field. The $ must appear prior to the comment.

Image: *EOF

Purpose: To denote the end of file or the end of this print or plot routine definition.

Format: *EOF - Required mnemonics for the end-of-file statement.

The print/plot routine definition subset is terminated by the name QUIT starting in column 1 and following the last *EOF. The QUIT card also represents the minimum data requirements for the subset.

Examples of output routine definition cards are shown in table 2-VIII.

When a Preprocessor-generated print routine is executed, the output will be printed with a title line consisting of the routine name and the current simulation time. The variables requested will be printed in a NAME-VALUE sequence, five to a line, in the order and type specified during input.

2.3.5 MAXMIN

This data subset allows the user to build a routine called MAXMIN through input data cards using the same card format used to define the print/plot routines. MAXMIN is a schedulable and/or callable routine designed to determine the maximum and minimum values for the variables specified. Upon completing the run and entering subroutine TERMIN, entry point MMPRNT in SELECT

TABLE 2-VIII.- SAMPLE PRINT/PLOT ROUTINE DEFINITION CARDS

Card column 1

```
PLOTR
      TIME    =  LECCOM(1)
      ERRORX  =  FSCOM(30)*RTD,E
      ERRORY  =  FSCOM(31)*RTD,E
      ERRORZ  =  FSCOM(32)*RTD,E
*EOF

ACCPRT
AVCAX  =  ACTVEC(85)  $AV CONT ACC X
AVCAY  =  ACTVEC(86)  $AV CONT ACC Y
AVCAZ  =  ACTVEC(87)  $AV CONT ACC Z
*EOF

PRTACC
      AVCAX  =  ACTVEC(85),E
      AVCAY  =  ACTVEC(86),E
      AVCAZ  =  ACTVEC(87),E
*EOF

MISC
$          SHOW MISCELLANEOUS CAPABILITY
TIME  = =  LECCOM(1)
D     = =  ENVCOM( 324)    ,I
I23456  =  FSCOM(    8)    ,B
JOB     =  FSCOM(    2)    ,B
CTABLE  =  LECCOM(   4)*2.0,E
I       =  LECCOM( 631)*10 ,I
J       =  LECCOM( 632)*10 ,B
A       =  ACTVEC(5)
B       =  RCSC(5)
L       =  MASPRC(5),E
M       =  GRAVC(5),E
Y       =  GGTORC(5)
BIG     =  FSC1(297)*.23456789
$          SHOW OPERATOR CAPABILITY
S1      =  IMUC(1)/-RTD
S2      =  RCSC(1)*-PTK
S3      =  ACSC(1)/-16.13
S4      =  ACCC(1)*-14.23763
S5      =  SLOSHC(20)+26.7
S6      =  BENDC(20)-0.0043
S7      =  FSCOM(20)*FSCOM(18)
S8      =  ACTVEC(20)/-ACSC(10)
*EOF
QUIT
```

is called to print the summary of the maximum and minimum values. MMPRNT can also be called or scheduled to print during the run. The generation of this routine is initiated when NAME is MAXMIN. This data set is also terminated by a *EOF.

When the Preprocessor-generated entry point MMPRNT is called, a summary print is made stating first the time of the print and then listing the variable names and the maximum and minimum values and the times they occurred.

2.4 SIMULATION DATA SET

This section defines the SOAP simulation data card sets.

2.4.1 SIMULATION TYPE

Tables 2-IX and 2-X illustrate the two types of SOAP simulations, initial and restart. The initial simulation is from cards only.

Card 1:

Image: TYPE S.F

Purpose: To define the SOAP simulation type. Do not add any type of comment to this card.

Format: Free-field format.

TYPE - Word INIT for an initial simulation and the word RESTAR for a restart simulation.

S.F - When TYPE = INIT, S.F is the initial simulation time in seconds. When this time is nonzero, the scheduling of all SOAP routines is shifted by this value.

When TYPE = RESTAR, S.F defines the particular state image to be used for restart. S is the state image number and F is the particular file on the state image file in which it is located.

2.4.2 RESTART SIMULATION

The restart simulation initialization consists of reading a state image (labeled COMMON images) stored on magnetic tape or mass storage from a previous simulation and modifying it by additional card input. No routines are read.

The state image input tape or file must be assigned to logical unit 7 for restart runs. Any new state images are output to logical unit 8.

TABLE 2-IX. - SAMPLE SOAP INITIAL INPUT DECK

Card	Column 1 ↓
1	INIT 40.0
2	LECCOM(956) = 3000 $ FSCOM INPUT AND ...
3	LECCOM(957) = 1000 $ ENVCOM INPUT AND ...
4	ACTVEH(163) = 1 $ ATTFLG FOR IC PASS
	(Additional input data cards)
5	*EOF
6	SI 1 $ SECOND TRAJECTORY (This is a new simulation)
7	FSCOM(384) = 1
8	FSCOM(1485) = 0.0
9	*EOF
10	NORMAL
11	*EOF

TABLE 2-X. - SAMPLE SOAP RESTART DECK

Card	Column 1 ↓
1	RESTAR 7.1
2	LECCOM(621) = 19 $ PLOTR1 LU=19
3	FSCOM(105) = PLOTR1 $ PLOTTER NAME
	(Additional input data cards)
4	FSCOM(115) = 0.5 $ DELTA T FOR PLOTR1
5	*EOF

When setting up a restart simulation, the user must ensure that the COMMON assignments and lengths in all the routines to be used in the restart are the same as those defined in the initial simulation. If they are, the same routines, new routines, or edited routines can be used.

Example: A sign error was found to exist in an atmospheric model equation for altitudes greater than h. If this model could be corrected without a COMMON block location change, a restart simulation could be made beginning at some altitude below h using the edited model.

Therefore, the preprocessor configuration specifications may be changed for a restarted run.

A complete set of preprocessor schedule cards must be defined for the restart simulation. This includes all before and after routines. The preprocessor uses these cards to generate the calling linkage in subroutine SELECT to all scheduled routines. In the typical case, the same set of schedule cards defined in the initial simulation is used. However, start times, stop times, delta times, and priorities may be changed. A new print routine can be defined by the preprocessor and used as a "before" and/or "after" scheduled routine for debug purposes.

If a simulation start time was specified on the initial simulation, then the user on the restart simulation must account for this time in the start and stop times of his restart schedule cards.

The input data cards for the initial simulation are not used. Only the data needed to change the run is required. If the reason for this restart run is a program change, data input is not required.

2.4.3 DATA INPUT

The data input cards for an initial or a restart simulation have the same format (see tables 2-IX and 2-X).

Cards 2, ..., 4:

Image: COMMON (INDEX) = DATA OP_1 SF_1 OP_2 SF_2 OP_3 SF_3 OP_4 SF_4 OP_5 SF_5

$ COMMENT

Purpose: To scale and input data into SSFS single and double precision COMMON blocks.

Format: Free-field format.

COMMON - Any one of the COMMON blocks defined in table 2-IV. When defining COMMON block lengths in input cards, the specification should appear before any input reference is made to the particular COMMON block. The COMMON block must be dimensioned in a routine equal to or greater than the specification being made.

INDEX - Location of the COMMON block to be initialized.

DATA - Integer or an octal, floating-point, or Hollerith number to be stored in the COMMON block. An integer is signified by the absence of a decimal point; it may not exceed 34,359,638,367. An octal number is signified by a B at the end of the number; it must be 12 digits or fewer. A floating-point number is signified by a decimal point; it is restricted to a card field of 24 columns or fewer and may be of normal or exponent form (e.g., 100.0, 1.0E+2, or 2.0D+3). A Hollerith number is signified by an initial alphabetic character; it must be six characters or fewer and is stored left-justified with blank fill.

OP_i - Optional operation code used to scale DATA prior to loading. For each OP_i used (a maximum of five is allowed), an SF_i must be supplied. OP_i is one of the four arithmetic

operators: +, -, *, or /. The evaluation of the input expression is in a strict left-to-right order, with all operators having the same weight.

SF_i - Input following an OP_i which is used to scale DATA prior to loading. SF must be of the same type as DATA. For numeric data, SF_i has the same restrictions imposed upon it that DATA does. SF_i may also be one of the 19 mnemonic conversion factors defined in table 2-VII. The same SF_i mnemonics are used for double precision. In this case, the system keys off of the COMMON block type.

$ - Optional code used to terminate an input card, leaving the rest of the card available for comments. If $ is placed in column 1, the entire card is considered to be a comment card and is centered on the page.

Card 5: The input data set must be terminated by an *EOF card.

Examples of data input cards are shown in table 2-XI. The number of inputs or input phrases per card is not restricted, but a data field may not be split between cards. Several data fields may be contained on one card (i.e., for loading into consecutive COMMON block locations) with the data fields separated by commas. Data input may be continued from one card to the next with the comma at the end of the statement. For loading into several consecutive locations, a particular location may be left unchanged by leaving a blank field on the card (i.e., two consecutive commas in adjacent columns or separated by blanks). The data set is terminated with an *EOF card. When specifying the length of a COMMON block, the input value must be an integer.

2.4.4 RECYCLE

A recycle is initiated during a simulation run by scheduling subroutine RECYCL or by a call to RECYCL by the flight software. Initiation of RECYCL results in a pause in the simulation run to (1) read input cards and continue the simulation from that point or (2) "roll back" to a previous point in the simulation run, read input cards, and continue the simulation from that point.

TABLE 2-XI. - SAMPLE INPUT DATA CARDS

CARD	FREE FIELD DATA INPUT CARD DEFINITION (Card columns 1 through 80)
1	$ PUT ANYTHING YOU WANT ON A COMMENT CARD
2	LECCOM (955) = 1000, 1000, 1000, 1000, 1000, 1000, 10, 10, 10,
3	10, 10, 10, 10, 10, 1000,
4	50, 50, 50, 50
5	F S C O M (1) = 1 $
6	ENVCOM (976) = 2, ENVCOM (977) = 2.0 $
7	FSCOM(1000) = 1000.0
8	FSCOM(2) = 2*2 $ INTEGER SCALE FACTOR
9	FSCOM(3) = 10B, 11B, 12B $ OCTAL LOAD
10	FSCOM(6) = 6.0 * 1.0 $ FLOATING POINT MULTIPLY
11	FSCOM(7) = 7.0E+01, 80.0E-1, 1.0*900.0E-2 $ EXPONENTS
12	FSCOM(11) = 12345678901B $ MAX OCTAL
13	FSCOM(12) = 99999999999. $ MAX FLOATING POINT
14	FSCOM(13) = 34359638367 $ MAX INTEGER
15	FSCOM(14) = -14.0, -7.5*2.0, +2.0*-8.0, -8.5*-2.0 $ MINUSES
16	FSCOM(18) = 1., 2., 3., 4., 5., 6., 7., 8., 9., 10., 11., 12., 13., 14., 15.
17	FSCOM(41) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
18	FSCOM(75) = 999999999., 999999999., 999999999.
19	FSCOM(80) = 0, FSCOM(81) = 1, FSCOM(82) = 2, FSCOM(83) = 3, FSCOM(84) = 4,
20	FSCOM(87) = -3*+2, 3*-3, -4*-5 $ INTEGER SIGNS

TABLE 2-XI.- CONCLUDED

CARD	FREE FIELD DATA INPUT CARD DEFINITION (Card columns 1 through 80)
21	FSCOM(100) = 1.0 * ITM $MNEMONICS
22	FSCOM(101) = 1.0 * FTM
23	FSCOM(102) = 1.0 * STK
24	ACTVEC(1) = 1.0, GGTORC(1) = 1.0, RCSC(1) = 1.0, MASPRC(1) = 1.0, GRAVC(1) = 1.0
25	FSCOM(402)= 2,,,,6,7,8,,10, ,12,13,14,FSCOM(440)=1.0,2.0,3.0,,,,7.0,,9.0
26	FSCOM(727) = 9.0*9.0+1.0-2.0/2.0*3.0 $ = 120.0
27	FSCOM(748) = 2*7*1*2*2, FSCOM(749) = 1+2+1+2+1 $56, 7
28	FSCOM(757) = 15*2+5-10/5*2 $ 10
29	FSCOM(760) = 5.0765432*17.6/4.5*FTM*DTR $
30	FSCOM(767) = 1.0/PTS/STP/IME/IEM/2.0 $ ABOUT 0.5
31	FSCOM(769) = ABCDEF $ HOLLERITH INPUT
32	FSCOM(770) = XYZ $ HOLLERITH INPUT
33	DPFSC(10) = 5.672D+4/17.3D+2
34	DPFSC(11) = 4.0D+2*FTM
35	DPFSC(12) = 5.0D+2,400.0,,0.732165D-4

The SSFS has three types of recycle capabilities (see card 6, table 2-IX): NORMAL, SI, and SISAVE.

a. The NORMAL recycle instructs the SOAP to read input cards and resume the simulation run from that point.

b. The SI recycle instructs the SOAP to read the state image specified on the SI input card. Input cards following the SI card are read, and the simulation run is continued from that point of the state image just read in. State image dumps that occur after the recycle are written over the old state images that follow the one used to reinitialize the simulation.

c. The SISAVE recycle instructs the SOAP to dump a final (for this trajectory) state image on the tape, write an end-of-file code, and "roll back" to the state image specified on the SISAVE card. The state image is read into core; the tape is repositioned past the end-of-file; and the simulation reads additional input cards, dumps another state image, and resumes the simulation from the point of reinitialization. State image tapes produced by the SISAVE capability contain one file for the original trajectory, plus additional files for each SISAVE recycle.

The SISAVE recycle capability was designed primarily to maintain a permanent record of a multiple trajectory run, so that any trajectory could be rerun to diagnose problems without having to repeat the complete run. The SI recycle is used more frequently for multiple trajectory runs, since most users are not concerned with saving a permanent record of the trajectories, and SI is more economical. The state image output file must be assigned to logical unit 8.

For multiple trajectory runs, a NORMAL recycle deck (NORMAL, *EOF) must follow the last SI or SISAVE recycle deck to terminate the run normally. This is required because an SI or SISAVE recycle causes a backup in time and causes RECYCL to be rescheduled when that time is reached again.

The optional recycle decks follow the simulation input deck shown in table 2-IX. All three types of recycles have the same input card structure or format.

Card 6:

Image: TYPE S

Purpose: To define the type of recycle.

Format: Free-field format.

TYPE - One of three words: NORMAL, SI, or SISAVE.

S - Code used only when TYPE is SI or SISAVE. S is an integer corresponding to the state image at which reinitialization is to occur.

Cards 7 and 8: Cards 7 and 8 have the same format and purpose as input data cards 2, ..., 4 in section 2.4.3.

Card 9: Each recycle data set is terminated with an *EOF card.

2.5 POSTPROCESSING

Postprocessing of the SOAP consists of plotting, using one of the two SSFS plotter packages: the production plotter or the high-speed plotter. The two plotters are denoted by their absolute element names, PLOTER and SPDPLT, respectively. With PLOTER, the user has the option of selecting CalComp, printer, and/or SD-4060 plots. With SPDPLT, the user can select CalComp and/or printer plots.

PLOTER or SPDPLT may be run after a simulation, in an in-line mode, or in a subsequent run to produce plots of data dumped onto a tape or mass storage file during the simulation.

2.5.1 PRODUCTION PLOTTER

With the production plotter (PLOTER), the user has the option of specifying the following:

a. One, two, or three variables on the same graph with a single abscissa parameter

b. Logical unit and file from which the data is to be plotted

c. Any combination of CalComp, SD-4060, or line-printer plots

d. Frequency of the data to be plotted

e. Number of points between symbols for CalComp plots for the set and/or for individual plots

f. Graph and axis titles

g. Tabular listings of the parameters to be plotted

h. Solid line or point plots

i. Number of frames (limited to four) in which the data is to be plotted and also whether multiple frames are to have the same scale

j. Standard CalComp scaling or an alternate SCALIE scaling routine, which may produce larger plots for the plot set

k. Axis scaling for an individual plot, which overrides the selection in the previous option listed

l. Plots from a portion of the input data file related to a specified time interval

m. Multiple plot sets with minimum input requirements

n. Choice of symbol representation for each variable

o. Symbols may be rotated to display an additional variable

The limitations in PLOTER include:

a. Maximum of 2000 points per plot for CalComp plots. Additional points are processed as separate plots.

b. Maximum of 200 points per frame for SD-4060 or printer plots. This is limited internally by recomputing the frequency of the points.

c. Maximum of four frames per plot.

2.5.2 HIGH-SPEED PLOTTER

The high-speed plotter (SPDPLT) has the same capabilities and limitations as PLOTER with the following exceptions:

a. No SD-4060 film plots

b. No optional graph and axis titles

c. One frame per plot request

d. No more than the first 1350 points from the input file plotted for CalComp plots

e. Limit of 100 prints internally for line printer plots

f. Maximum of 30 plot variables

g. No user-defined input scaling, although selection of SCALE or SCALIE for the plot set is permitted

h. No capability to plot a time interval portion of the data file

2.5.3 PLOTTER INPUT

Despite all the options available in the plotters, the programs are designed to require a minimum of input. In the simple plotter input deck shown in table 2-XII, the plot request set control card contains only the plot identification field, while the individual plot request cards list only the variable names. This type of deck is most often submitted and results in valid sets of printer and CalComp plots using standard options and nominal data values. For example, the blank plot request set control card implies that the data to be plotted is to be obtained from a mass storage file on file 1 of assigned logical unit 20. Every point is to be plotted to produce CalComp and line printer plots, using the SCALE routine for axis scaling. Using this information, plots are then to be processed for variables defined on the individual plot requests.

TABLE 2-XII. - SAMPLE PLOTTER INPUT DECK 1

Card	Col. 1-	Col. 10-15	Col. 20-25	Col. 30-35	Col. 40-45
1	PLT01				
2		TIME	AIRSPD		
3		TIME	AVACX	AVACY	AVACZ
4		LAT	LONG		
		(Additional input cards)			
5	*EOF				

Additional input is required when nonstandard options are desired, as shown in the sample input deck in table 2-XIII.

The input sets for the two plotters are compatible, with the high-speed plotter (SPDPLT) ignoring some of the options. The following sections describe the input cards for the production plotter (PLOTER); SPDPLT limitations are noted.

TABLE 2-XIII.- SAMPLE PLOTTER INPUT DECK 2

Card	Col. 1-6	Col. 7-15	Col. 17-25	Col. 29-35	Col. 40-60
1	PLT02	2 10		*	
2		10.0	80.0		
3		*TIME	*ATAK	*ATAKC	
4	S	10.0	80.0		4.0 14.0
5		TIME			
6		TIME	/AIRSPD		
7	*	TIME	POS		
8		1 ANG			
			(Additional input cards)		
		LONG	LAT		
	G LONGITUDE VS. LATITUDE				
	A LONGITUDE(DEGREES)				LATITUDE(DEGREES)
	*EOF				
		3	*		
		TIME	AVACX	AVACY	AVACZ
	*EOF				
	*EOF				

2.5.3.1 Plot Request Set Control Card

Card 1 (table 2-XIII):

Image: IDPLT IFILE NPTS IFREQ NSET LU ISTORE IPART NOCAL NOPRT SD4060 ISCALE

Purpose: To define the general input/output requirements for the following plot request set.

Format: Fixed-field.

Card column	Name	Format
1-5	IDPLT	A5
6-10	IFILE	I5
11-15	NPTS	I5
16-20	IFREQ	I5
25	NSET	A1
29-30	LU	I2
31	ISTORE	A1
35	IPART	A1
40-45	NOCAL	A6
50-55	NOPRT	A6
60-65	SD4060	A6
80	ISCALE	A1

IDPLT - Plot identification. If blank, no plot ID will appear on plots or printed output.

IFILE - Defines the file from which the data is to be plotted for the following plot request set. If blank or 0, file 1 is assumed.

NPTS - Is applicable only to CalComp plots. It specifies the number of data points between symbols for the set. If blank, it is assumed to be 50. This option may be overridden on individual plot requests.

IFREQ - Defines the frequency at which the data for the following plot request set is to be plotted. If IFREQ is blank, 0, or 1, all data is plotted; if 2, every other data point is plotted; if 3, every third data point is plotted; and so on. SPDPLT ignores this field.

NSET - If blank, individual plot request cards are required for all plots in the set, and this set of plot request cards becomes a standard set. If nonblank, the last standard set of plot variables is plotted as specified on this plot request card. Additional plot request cards may be supplied to supplement the standard set of plot requests for this set of plots only.

NU - Specifies the unit that contains the data to be plotted for the following request set. If blank or 0, logical unit 20 is assumed.

ISTORE - If nonblank, the data to be plotted is obtained from tape. If blank, the data is obtained from the mass storage file.

IPART - If nonblank, only the time interval portion of the plot data file is plotted, and a second set control card is required to specify the minimum and maximum plot times. This option is not valid in SPDPLT.

NOCAL - If blank, a CalComp plot is produced for each plot in the succeeding plot request set. If nonblank, no CalComp plots are produced unless one of the plot requests in the succeeding set indicates otherwise.

NOPRT - If blank, a line printer plot is produced for each plot in the succeeding plot request set. If nonblank, no line printer plots are produced unless one of the plot requests in the succeeding set indicates otherwise.

SD4060 - If nonblank, an SD-4060 microfilm plot is produced for each plot in the following plot request set. If blank, no microfilm plots are produced unless one of the plot requests in the following set indicates otherwise. This field is ignored by SPDPLT.

ISCALE - If blank, the standard CalComp SCALE routine is used for the plots. If nonblank, the SCALIE routine is used for axis scaling. SCALIE produces larger curves than SCALE, but the graph scale may not be as easy to interpolate. This option may be overriden in PLOTER (but not in SPDPLT) by user-defined scaling for individual plots.

2.5.3.2 Optional Plot Set Control Card

Card 2 (table 2-XIII): This card is supplied only when column 35 of card 1 is nonblank.

Image: TMIN TMAX

Purpose: To define a limited time interval for the plot request set.

Format: Fixed-field.

Card column	Name	Format
1-10	TMIN	F10.1
11-20	TMAX	F10.1

TMIN - Minimum plot time; may be greater than or equal to the initial simulation time.

TMAX - Maximum plot time; may be less than or equal to the simulation termination time.

SPDPLT reads but ignores this card when column 35 on card 1 is nonblank.

When this option is used in PLOTER, the following conditions must be met:

a. TIME must be the first variable in the plot dump file.

b. Logical unit 29 must be assigned as shown in table 2-II (card 15).

2.5.3.3 Individual Plot Request Cards

Card 3 (table 2-XIII):

Image: A I X I YA I YB I YC XAXS YAXS NSYM NF FS C P S

Purpose: To define a plot to be produced.

Format: Fixed-field.

Card column	Name	Format
2	A	A1
9	I	A1
10-15	X	A6
19	I	A1
20-25	YA	A6
29	I	A1
30-35	YB	A6
39	I	A1
40-45	YC	A6
57-60	XAXS	F4.1
62-65	YAXS	F4.1
67-70	NSYM	I4
72-73	NF	I2
75-76	FS	A2
78	C	A1
79	P	A1
80	S	A1

A - For PLOTER, if A is an asterisk (*), then there is another card immediately following with format as specified in the description of card 7 following, which gives data determining what symbol is to represent a certain variable and at what angle the symbol is to be printed.

I - For PLOTER, if I is an asterisk (*), a tabular listing of the data plotted is produced for the variable which follows the asterisk. For SPDPLT, if any I is an asterisk, a listing is produced for all variables specified on the plot request card. For both plotters, if the I preceding YA, YB, or YC is a slash (/), the corresponding plot (curve) consists of an X at each point instead of the normal line connecting the points.

X - Name of the X-axis variable.

YA - Name of the Y-axis variable.

YB - Optional name of the second Y-axis variable.

YC - Optional name of the third Y-axis variable.

XAXS - Applicable only to CalComp plots. It defines the X-axis plot frame length in inches. If blank, 7 inches are assumed.

YAXS - Applicable only to CalComp plots. It defines the Y-axis plot frame length in inches. If blank, 5 inches are assumed. YAXS is limited to 9 inches.

NSYM - Applicable only to CalComp plots. This code specifies the number of data points between symbols. If blank, it is assumed to be NPTS, as specified in columns 11 through 15 of card 1 of the data set.

(NOTE: The remaining inputs are ignored by SPDPLT.)

NF - Specifies the number of frames in which the data is to be plotted. If blank, it is assumed to be 1. NF is limited to 4.

FS - Used only if NF is greater than 1. If nonblank, it specified that all frames have the same maximum and minimum Y-axis values. If blank, the scales for each frame are independent of the other frames.

C - If nonblank, a CalComp plot is produced for this plot in addition to the types requested on the plotter control card (card 1). If blank, the plotter control card specifies the types of plots to be produced for this request.

P - If nonblank, a printer plot is produced for this plot in addition to the types requested on the plotter control card (card 1). If blank, the plotter control card specifies the types of plots to be produced for this request.

S - If nonblank, an SD-4060 plot is produced for this plot in addition to the types requested on the plotter control card (card 1). If blank, the plotter control card specifies the types of plots to be produced for this request.

Card 7 (table 2-XIII):

Image: ISYAANG ISYBANG ISYCANG

Purpose: To allow user specification of symbol to represent any of three Y variables (see below for choice of symbols)

Format: Fixed-field

Card column	Name	Format
9	IS	I1
10-15	YAANG	A6
19	IS	I1
20-25	YBANG	A6
29	IS	I1
30-35	YCANG	A6

IS - Integer defining choice of symbol

YAANG - Name of variable angle at which the first Y-axis variable symbol is drawn.

YBANG - Same as YAANG for second Y-axis symbol.

YCANG - Same as YAANG for third Y-axis symbol.

2.5.3.4 Optional Plot Request Cards

Any combination of the optional cards defined in this section may be inserted immediately following an individual plot request in the PLOTER input deck.

Axis scaling card:

Image: S XMIN XMAX YMIN YMAX
or
S XMIN DELX YMIN DELY

Purpose: To permit user input of axis scaling parameters. This card is ignored by SPDPLT.

Format: Fixed-field.

Card column	Name	Format
1	S	A1
2-12	XMIN	E11.1
13-24	XMAX	E12.1
	or	
25-36	DELX	E12.1

Card column	Name	Format
37-48	YMIN	E12.1
49-60	YMAX	E12.1
	or	
61-72	DELY	E12.1

S - Character S, which must appear in column 1 of the axis scaling card

XMIN - X-axis minimum value

XMAX - X-axis maximum value

or

DELX - X-axis units per inch of plot length

YMIN - Y-axis minimum value

YMAX - Y-axis maximum value

or

DELY - Y-axis units per inch of plot height

When an S appears in column 1, minimum values of X and Y must be specified. Either maximum values of X and Y or unit increments per inch on the X and Y axes must be specified consistent with the dimensions specified in XAXS and YAXS of the plot request card.

Plot title card:

Image: G LONGITUDE VS. LATITUDE

Purpose: To permit the specification of a title for a plot. This title card is applicable to the plot request card immediately preceding this card (i.e., card 4 in table 2-XII. This card is optional for PLOTER and is ignored by SPDPLT.

Format: Fixed-field.

Card column	Name	Format
1	G	A1
2-80	TITLE	13A6,A1

G - Character G, which must appear in column 1 of the plot title card.

TITLE - Title of the plot.

Axis title cards:

Image: A LONGITUDE(DEGREES) LATITUDE(DEGREES)

Purpose: To permit the specification of a title for the X and Y axes of the plot. Two axis title cards may be used to define an X-axis and up to three Y-axes. The axis title card is applicable to the plot request card immediately preceding it. The X-axis title is defined on the first half of the axis title card. The first Y-axis title is defined on the last half of the card. If two or three Y-axis variables are defined, a second axis title card is permitted, with the first half of the card defining the second Y-axis parameter and the second half defining the third Y-axis parameter. This card is optional for PLOTER and is ignored by SPDPLT.

Format: Fixed-field.

Card column	Name	Format
1	A	A1
2-37	AXIS1	6A6
41-76	AXIS2	6A6

A - Character A, which must appear in column 1 of the axis title card.

AXIS1 - Title for the X-axis or the second Y-axis.

AXIS2 - Title for the first or third Y-axis.

End-of-file card:

Image: *EOF

Purpose: To denote the end of file or the end of a data set within a data stream.

Format: *EOF - Required mnemonics for the end-of-file card.

Any number of plot request sets may be inserted in the input deck, with an end-of-file card between each set. Two end-of-file cards should follow the last data set.

2.5.3.5 Logical Unit Assignments

For most plotter runs, logical units 3, 14, 20, and 29 must be assigned by inserting corresponding @ASG and @USE statements in the input deck. Table 2-I describes the input/output logical unit requirements for the plotters.

2.5.3.6 Diagnostics

If one or more of the dependent variables requested as YA, YB, and YC are not defined in the plot dump routine (section 2.5), the undefined variables cannot be plotted, but a message is printed at the bottom of the plot frame. For example,

PP IS NOT DEFINED PLOT IGNORED

If the independent variable is undefined in the plot dump routine, then a blank frame is printed with the following message at the bottom:

INDEPENDENT VARIABLE IS NOT DEFINED PLOT SKIPPED

2.5.3.7 Plot Output Data Format

There is a requirement for the SOAP to generate a data file or tape to be used as input to the SOAP plot processors and/or other programs. Through the SOAP Preprocessor scheduling and plot routine generation capability, it is possible to output any information stored in labeled COMMON blocks at the scheduled plot delta time.

The SOAP plot output data is written using the standard computer system format onto a file or tape in the following manner:

a. Symbol record

b. Data records

c. EOF records

The symbol record consists of the variable names as assigned in the SOAP Preprocessor plot routine data set and the number of plot variables (N). The symbol record is 256 words long and is arranged in the following manner:

a. Locations 1 through N (where $N \leq 254$) - variable names assigned to plot data values.

b. Location N + 1 - number of variables (N) in the plot data set.

c. Location 256 - Number of variables (N) in the plot data set.

The data records consist of the values for the plot data set. These records are N words long and are written out each time the plot routine is executed.

The EOF record is 256 words long and is used as a file mark. It consists of a coded end-of-file (left-justified Hollerith data - EOFbbb) in the first word. The values in the other 255 words are not important. The SSFS has multifile capability, and each file is terminated by an EOF record. Two EOF records are written at the end of the last file. It is necessary that this last record be a maximum of 256 words long because of the variable lengths possible for the Data records.

Table 2-XIV is a sample format of the plot output data and table 2-XV is the coding necessary to read the plot output data if used as input to another program.

It should be noted that if an SSFS simulation is made with the plot output data going to a file and later copied to tape, the tape cannot be read directly. It would be necessary to copy the tape back to a file, which can

TABLE 2-XIV. - PLOT FILE FORMAT

WORD / RECORD	1	2	3	4	5	256
1	TIMEⱭⱭ	RXⱭⱭⱭⱭ	RYⱭⱭⱭⱭ	RZⱭⱭⱭⱭ	4	4
2	0.0	6.5E+6	0.0	0.0		
3	10.0	6.4E+6	1.0E+5	-1.0E+5		
i-2	490.0	-4.1E+5	3.1E+6	-3.1E+6		
i-1	500.0	-4.2E+5	3.2E+6	-3.2E+6		
i	EOFⱭⱭⱭ					

TABLE 2-XV. - CODE TO READ PLOT FILE

```
      DIMENSION BUFF(256), IBUFF(256)
      EQUIVALENCE (BUFF(1), IBUFF(1))
      DATA IEOF/3HEOF/,LU/20/

C                          READ SYMBOL RECORD
      READ (LU) IBUFF
      NDP = IBUFF(256)
C                          READ DATA RECORD
  100 READ (LU) (BUFF(I), I=1, NDP)
C                          IS THIS EOF RECORD
      IF (IBUFF(1).EQ.IEOF) GO TO 200
C
C     *** USE DATA ***
      GO TO 100
C
C     *** ALL DATA READ ***
  200 CONTINUE
```

then be read. If the data goes directly to tape during a simulation, the data can be read directly from tape.

Also, any tape or file generated by other programs in the above format could use the SOAP plot capabilities.

3. UTILITY ROUTINES

These routines comprise SOAP preprocessor, executive, and postprocessor routines, and general purpose routines which the user may include in a flight control model. Typically the user need not be at all concerned with the systems features and need only select those items from sections 4 and 5 which are desired. The only caution which must be included in this recommendation is that some mathematical functions have both single precision and double-precision versions. The double-precision versions support double-precision environment routines, while the single-precision versions may be used in the same simulation by flight control routines. The user need only be aware of the distinction and should not attempt to substitute.

3.1 PREPROCESSOR ROUTINES

The preprocessor writes the routines SELECT, CALMOD, and DUMMYS based on the environment input cards. It creates the appropriate wait list based on schedule cards and on print/plot requests. Typically, the user will neither call nor modify these routines directly, but will simply use the Preprocessor in accord with the runstream instructions in section 2.

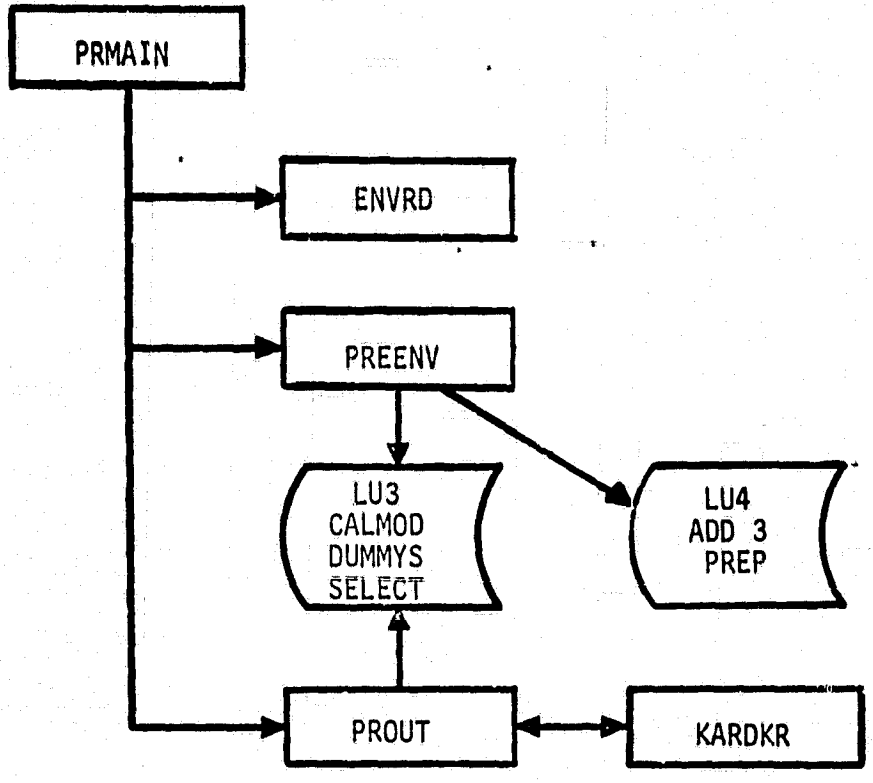
Preprocessor routines are:

ENVRD
KARDKR
PREENV
PRMAIN
PROUT

A functional diagram is given in figure 3-1 and a discussion of each routine follows.

ENVRD

Subroutine ENVRD first checks to see if the optional input card for controlling the listings of the FORTRAN subroutines (CALMOD, DUMMYS, SELECT) generated during preprocessor execution. If it is, the appropriate flags are set to obtain the desired listings. By default the most abbreviated listing of the preprocessor-generated FORTRAN subroutines are given. Subroutine ENVRD then reads and decodes a set of free-field input cards specifying the Environment



PRMAIN - Preprocessor MAIN program.

ENVRD - Reads and decodes the Environment configuration selection cards.

PREENV - Matches Environment model selection with desired configuration. The FORTRAN subroutines CALMOD and DUMMYS are generated on logical unit 3 (LU3). PREENV writes compilation and editing instructions for CALMOD, etc., on LU4.

PROUT - Builds FORTRAN subroutine SELECT from the schedule and print/plot data sets, and writes SELECT on LU3.

KARDKR - Decodes print/plot data cards for subroutine PROUT.

Figure 3-1.- Preprocessor functional diagram.

models to be used. The cards consist of phrases of the form:

MODEL = FILE.VERSION

where

MODEL is the name of an Environment function (e.g., ACTVEH). The element specified is typically the location of relocatable code for a driver program.

FILE is the name of the program file

VERSION is the version of the particular Environment model to be used to perform that function (e.g., AVEH1)

One input phrase to ENVRD is unique. This phrase selects the COMMON block definition routine (DCOM =). In processing the input, ENVRD stores the desired Environment configuration specified into two arrays with the object code names and versions (the object code name is of the form "NAME/VERSION") of the particular Environment model subroutines specified. After processing the input cards, ENVRD returns control to PRMAIN.

KARDKR

Subroutine KARDKR is called by PROUT to process each output routine variable definition card. The free-field cards (one per variable) specify a COMMON block name, a location within the COMMON block, a Hollerith name for the variable and an optional format and scale factor. KARDKR decodes the Hollerith card image and returns the information to PROUT. This includes identifying mnemonic scale factors, if specified. Whenever a bad card is detected, KARDKR sets a flag for PROUT.

PREENV

PREENV processes the input configuration cards decoded by ENVRD specifying the particular Environment model subroutines which are to be used during the simulation. During a simulation job, a secured file containing Environment model subroutines is assigned to the run. This file (designated by the single letter W) may contain several different subroutines performing the same Environment function and therefore being referenced by the same name. PREENV makes it possible for the computer system allocator to choose the correct Environment model subroutine to load into core. The environment model specified

calls any number of differently named subroutines, one for each vehicle. PREENV begins writing FORTRAN editing and compilation directives on LU4. These directives will compile the FORTRAN routines (CALMOD, DUMMYS, and SELECT) generated on LU3. The routines are first loaded using the ED processor and then compiled. This sequence allows the SOAP to be executed interactive from the terminal or in a batch mode. PREENV then uses the information supplied by ENVRD to begin writing an allocation element onto LU4. This portion of the element will supply the allocator with the necessary instructions to call the desired Environment model subroutines and their associated Block Data subprograms into core for the simulation. PREENV then writes the FORTRAN code onto LU3 for subroutines CALMOD and DUMMYS. PREENV then returns control to PRMAIN.

PRMAIN

PRMAIN is the main program which controls the flow of Preprocessor execution. Initially, it calls subroutine ENVRD for the selection of the particular Environment model subroutines which are to be used during the simulation. It then calls PREENV to define the Environment load directives from the input decoded by ENVRD and to build two routines for Environment interface. After returning, PRMAIN calls PROUT, which calls KARDKR, to build a routine for flight control routine scheduling and printing and/or plotting of requested simulation parameters. After calling PROUT, PRMAIN prints the duration (in seconds) of preprocessor execution and terminates.

PROUT

PROUT processes three sets of Preprocessor input cards.

a. Schedule data cards specify the names of the flight software routines and output routines to be executed during the simulation. Accompanying each routine name is a priority, delta time, start time, and stop time. There may also be specified the name of a routine to be executed before or after the scheduled routines call, with start and stop times. Using

the schedule data cards, PROUT builds a table of data to be used later as input (via data statements) to the SOAP Executive.

b. The print and plot output routine data cards provide the names of output routines which the Preprocessor is to build and define the variables to be output during the execution of each routine.

c. The maximum/minimum routine data cards provide the name of the output routine (MAXMIN) which the Preprocessor is to build and define the variables to be used during the execution of the routine. An entry point is generated when using this routine which is called at the end of the simulation to print a summary of maximum/minimum values of the variables specified.

As output, PROUT writes the FORTRAN subroutine SELECT onto LU3. PROUT begins by reading the schedule data cards and storing the information. In the process, PROUT replaces all stop times and delta times which are less than or equal to zero with values of 10^{10} and 10^{37}, respectively, to prevent looping during the simulation.

After finishing (reading an end-of-file (EOF) card) the schedule cards, PROUT processes the output routine data cards. These cards occur in groups, each of which contains an output routine name card, followed by one or more variable definition cards, and ending with the EOF card. PROUT uses the input cards to generate several arrays which later will be used in writing portions of SELECT. The processing of each variable definition card is accomplished by a call to KARDKR. The end of the output routine data cards is signified by the word "QUIT" in columns one through four of an input card. After this card has been read, PROUT begins the writing of subroutine SELECT onto LU3. SELECT contains branching logic used to transfer control to the first routine on the wait list during simulator execution and all output routines specified by the Preprocessor input, each routine having its own entry point within SELECT. If one of the words PLOTR, PLOTR1, PLOTR2, ..., PLOTR8, or PLOTR9 appear as an output routine name (signifying a plot dump routine), a call to PLTOUT is inserted in the SELECT logic for plot dumps. If the output routine name specified is MAXMIN, the maximum/minimum logic is inserted in the SELECT logic. For all print routine names, a call to a DATOUT is included in the output routine logic for the actual printing. After writing SELECT, PROUT returns control to PRMAIN.

3.2 SOAP EXECUTIVE SYSTEM

Figures 3-2, 3-3, and 3-4 depict SOAP operation. Commands are carried out, flight control routines are executed, and environmental information is updated when the time designated for such an operation and the simulation time coincide. The wait list and command file hold lists of operations to be performed and commands to be executed, respectively. As each event takes place, it is removed from the appropriate list. In order to cause an entry to be made in the command file, CMDFIL is called by the user. To cause a routine to be scheduled or rescheduled, calls to FREQEJ (or FREQUP), CHGBLD, REWTLT, or SCHED are used.

The user will not have a need to call most of these routines directly. The exceptions include those routines mentioned above, ENDJOB (to remove a routine from the wait list), and routines such as TERMIN, SIPRINT, and SIDUMP. TERMIN ends simulation execution and outputs lists of all COMMON block locations. SIPRINT causes a state image, i.e., COMMON block location printout. SIDUMP is similar to SIPRINT but a variety of output devices are available.

When constructing flight control routines, the user must remember that any routine which is to be scheduled for execution at regular intervals must be identified to the system by a schedule card (see section 2). After execution of such a routine, it must be rescheduled by a call to FREQEJ or FREQUP within the subroutine. Routines which are only called by other routines have no special requirements.

The input/output routines INPUT and PINOUT have versions designed to deal with double-precision blocks. The names are the same, the difference is specified in the mapping stage of execution. Listed below are the names of subroutines involved in the SOAP executive; descriptions follow.

BLKPLT	DUMMYS	FREQEJ	ISPPRT	PLTOUT	RESTRT	SORT3V
CALENV	ENVC	FREQUP	JSPPRT	PNNOUT	REWTLT	SREAD
CALMOD	ENVCAL	HDLNS	KALNOW	PRTSDT	SANDF	SUPTIM
CHGBLD	ENDJOB	HEDLNP	MAIN	PRWAIT	SCHED	SWRITE
CMDEXE	ENVSKL	INOUT	NEXTIM	QZKILL	SELECT	SYSPRT
CMDFIL	EXECT	INPUT	NINOUT	RECOV	SIDUMP	TERMIN
DPOUT	EXECTR	INPUT2	PINOUT	RECYCL	SIPRIN	TIMER

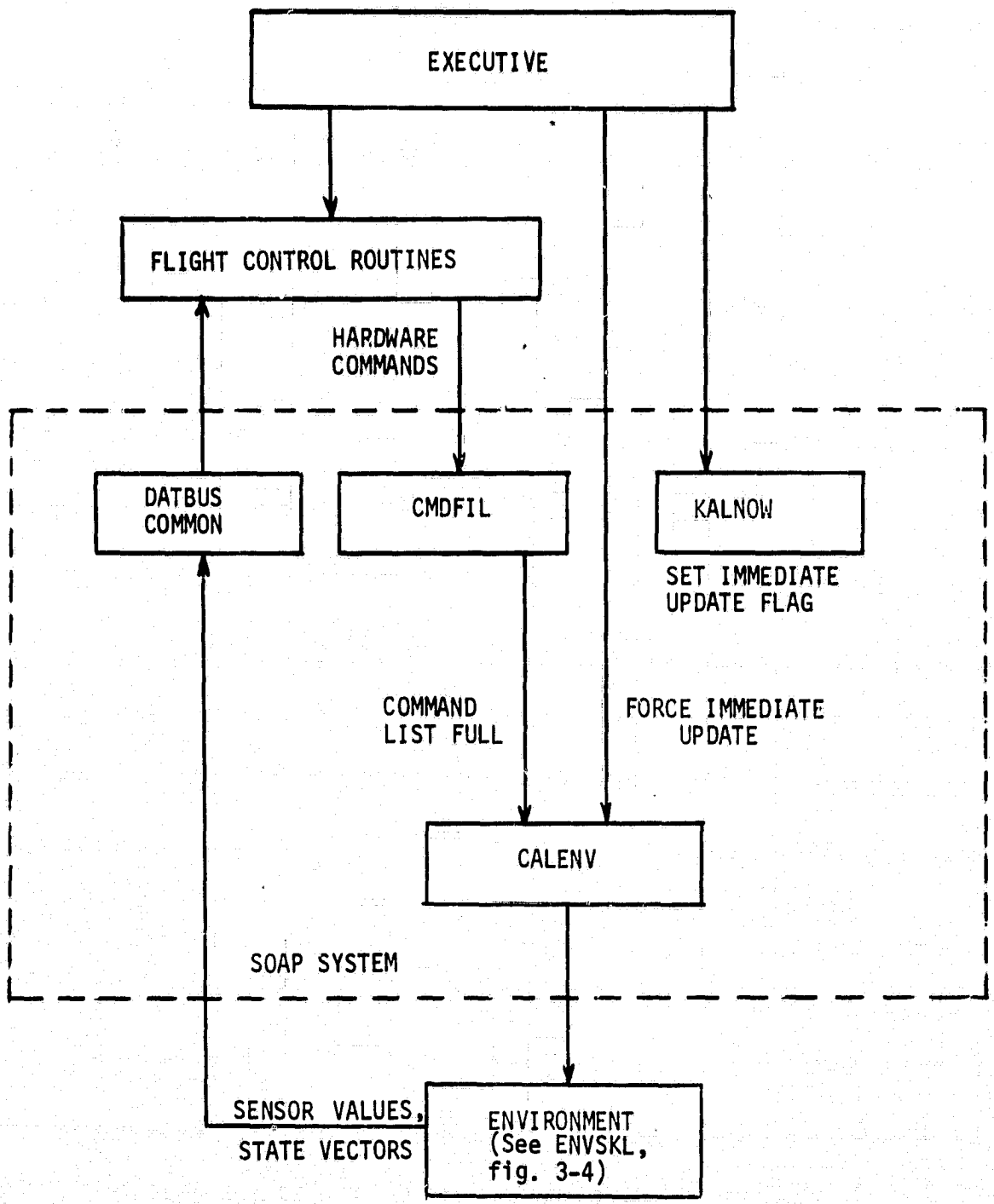


Figure 3-2.- Relationship of SOAP operational blocks.

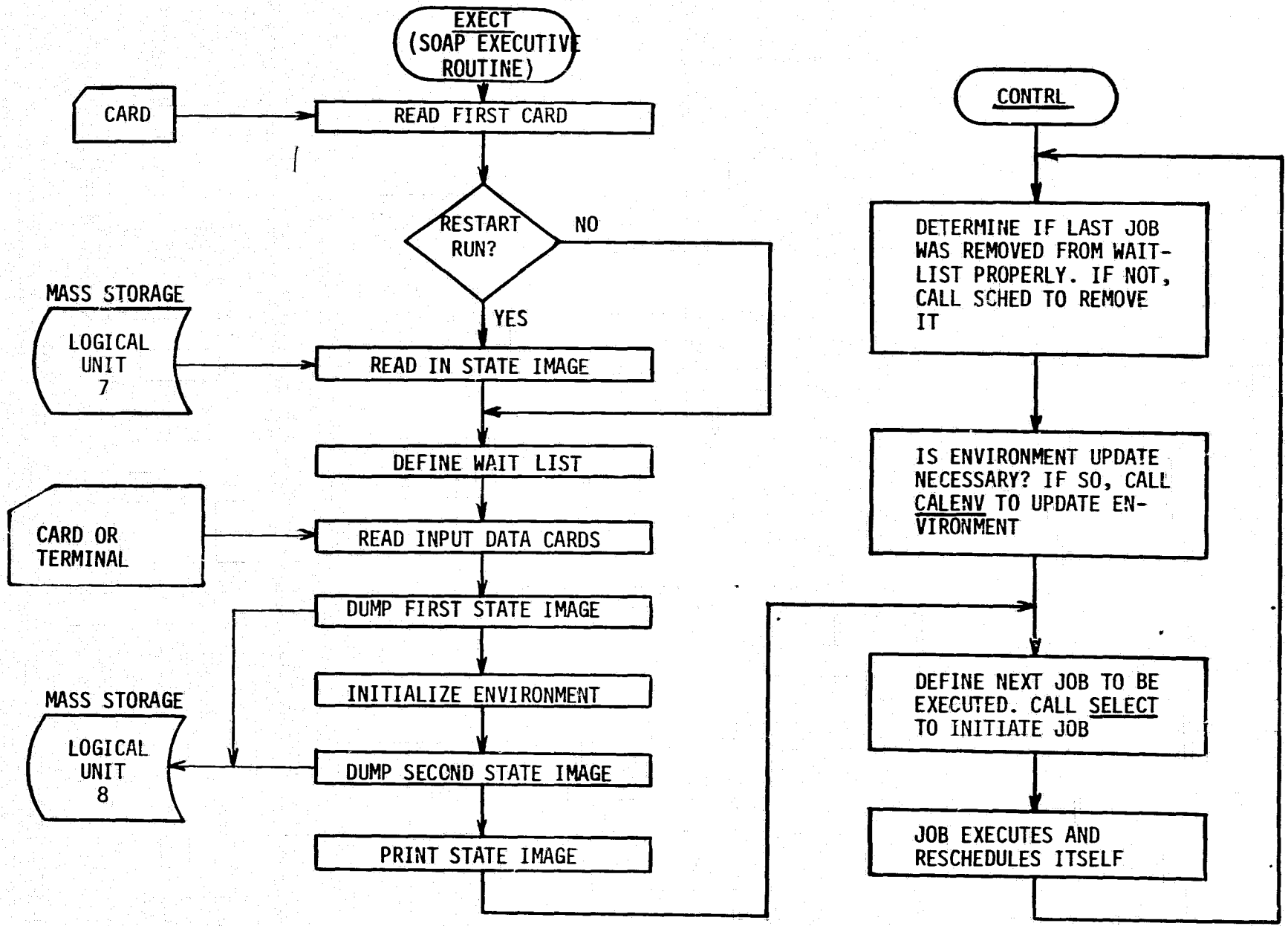


Figure 3-3.- Executive functional diagram.
(subroutine names are underlined)

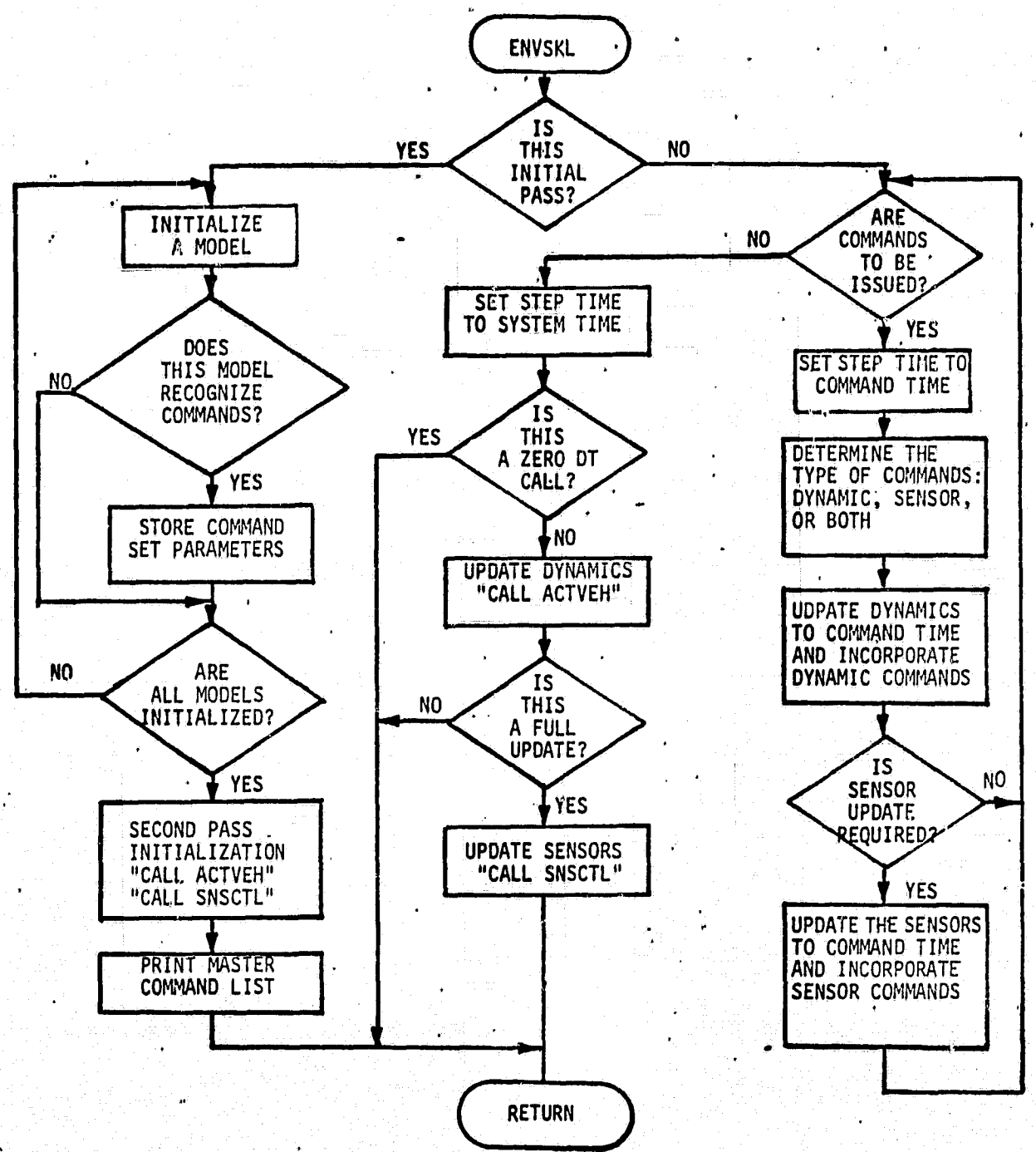


Figure 3-4.- ENVSKL functional diagram.

BLKPLT

Subroutine BLKPLT (NPOINT, NDATA, IBCD, ARRAY)

BLKPLT may be called by the flight software to dump one file of data for subsequent plotting. The data to be plotted must be stored in ARRAY, dimensioned ARRAY (NPOINT, NDATA). ARRAY thus contains NPOINT data points for NDATA variables. IBCD is an array of length NDATA containing BCD (e.g., TIME) names corresponding to the variables to be plotted. These names will be used by the plot processor to identify requested plots. After dumping the requested data, BLKPLT writes an EOF. BLKPLT uses LU20 for output.

CALENV

CALENV is used by all areas of the simulator for updating the Environment. Scheduling or calling CALENV is the only way in which the Environment can properly be updated. If an update is needed but not immediately necessary, LECCOM(704) may be set to a 1 (integer). The Executive will then update the Environment (via CALENV) before the next flight control routine is executed.

If an immediate update is required, the flight control should call CALENV directly. In either case, LECCOM(869) should be set to a 1 (floating-point) if a full Environment update is desired. Normally, the Environment update consists of only the active vehicle dynamics and the dynamics sensors. With LECCOM(869) set to a 1, all of the Environment is updated, including the non-dynamic sensors.

If CALENV is entered and the Environment has previously been updated to the current time and there are no commands to be incorporated, control is returned to the calling program.

CALMOD

CALMOD makes direct calls to the Environment model routines.

CHGBLD

Subroutine CHGBLD (JOBNAM, PRIOR, DELTAT, TSTOP)

CHGBLD may be called by the flight software to alter the Preprocessor data sets for scheduled routines. In this manner, the flight control can define its own future scheduling so that subsequent calls to FREQUP, FREQEJ, SANDF, or WLINFO will use the new data. JOBNAM (six-character or less binary code decimal (BCD) name) identifies the routine whose data set is to be altered.

CMDEXE

Subroutine CMDEXE handles the command execution function. Its operation is shown functionally in figure 3-5. Actually, there is logic in both ENVSKL and CMDEXE (not shown in functional flowcharts) for separate calls to CMDEXE for Dynamics commands and for Sensor commands.

CMDFIL

Subroutine CMDFIL (CODE, DATA)

CMDFIL is called by the flight control to file away commands for later execution by the Environment. The commands represent the flight computer output interface with the flight hardware. For the SOAP, the commands may range from a simulation of output channel bit setting (the most detailed simulation) up to the most function level (e.g., a thrust pointing vector).

CMDFIL is called with two arguments, CODE and DATA. CODE is the actual command code recognized by the Environment. DATA is a one-, two-, three- piece set of data representing the information associated with CODE. DATA might be a 1 or a 0 for a bit on or off, or it might represent a pointing vector. Both CODE and DATA are defined for each Environment model.

When called, CMDFIL first checks to ensure that it has room to file the command. At the present time, a maximum of 49 commands may be accumulated. If the table is full, CALENV is called to update the Environment and to incorporate any commands on the list scheduled to be executed before the current flight software time. This will make room on the push list for the new command. Before the

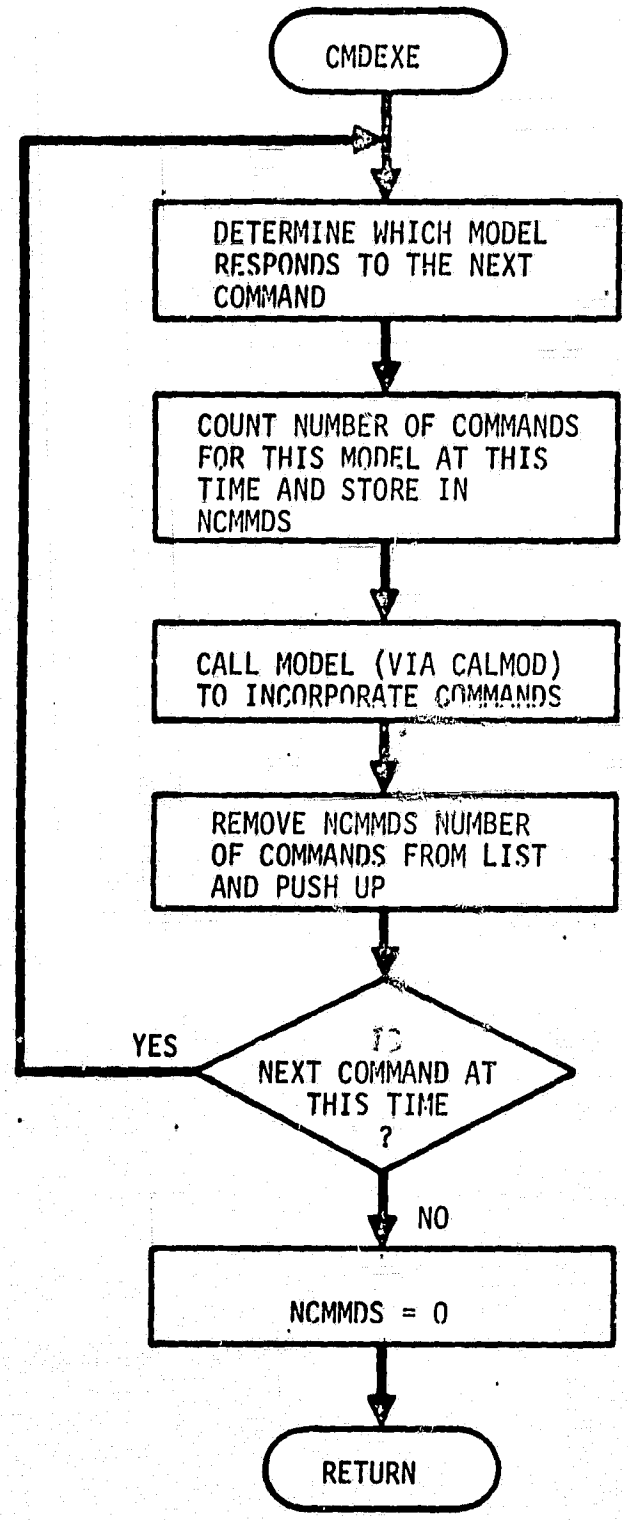


Figure 3-5.- Command execution functional diagram.

command is filed on the push list, any time delays associated with the command are added to determine the command execution time. The time delay is defined by the Environment, along with a flagword. The flagword specifies whether the command invalidates further Environment sensor extrapolation and whether an immediate Environment update is required by the issuance of the command.

If an immediate update is flagged, it is to be forced at the command execution time. The routine KALNOW, which sets the update flag, LECCOM(704), to 1 is scheduled at the highest priority at the command execution time. The correspondence of command codes with models used in the SOAP can be found in table 1-I.

DPOUT

Subroutine DPOUT (DPA, NF, NL, NDPCB)

DPOUT is a double-precision print output routine used to print the double-precision named NDPCB array DPA from location NF to NL in the double precision common block named NDPCB.

DUMMYS

DUMMYS supplies entry points with names corresponding to all Environment models which are not specifically required for the simulation at hand.

ENVC

Subroutine ENVC is an entry point in ENVCAL and performs identically to ENVCAL.

ENVCAL

Subroutine ENVCAL is a schedulable routine which when called performs an immediate environment update by calling subroutine CALENV.

ENDJOB

Subroutine ENDJOB is an entry point in SELECT and is used to terminate the execution of flight software and to return simulation control to CONTRL. This special termination routine is used to ensure that CONTRL regains control after the flight software execution, since the normal FORTRAN subroutine

linkages (using the RETURN statement) are sometimes destroyed during a simulation. All flight control routines which are scheduled or which complete the execution of a scheduled routine should call ENDJOB or FREQEJ, which calls ENDJOB.

ENVSKL

Subroutine ENVSKL is the Environment Executive routine and is the central control routine. Figure 3-4 represents a functional flow of ENVSKL. At the start of a simulation run (initialization pass), ENVSKL calls each of the functions which may be modeled. If a particular function is not modeled in a run, its call will be answered by a dummy routine supplied by the Environment system.

On subsequent calls, ENVSKL updates the Dynamic models, and if the full update flag, REASON, is set to 1, it also updates the Sensor models. The update continues, stopping to incorporate scheduled commands, until Environment time is equal to the flight control time.

EXECT.

EXECT is called by the MAIN program of the SOAP and accomplishes the simulator initialization. Initialization may involve a complete simulator initialization or a restart run (i.e., a resumption of an earlier run). EXECT calls INPUT to read the SOAP input cards, CALENV to initialize the Environment, SIDUMP for the optional initial State Image dumps, and SIPRIN for an initial State Image print for diagnostic purposes. After initialization, EXECT returns control to MAIN, which calls CONTRL for the actual simulation. EXECT should never be called by the flight control routines.

EXECTR

EXECTR is an entry point in CONTRL and is called first by MAIN to start the simulation. EXECTR defines the first job and the time of the first job, and calls SELECT to execute the job.

FREQEJ

Subroutine FREQEJ is an entry point in FREQUP and is used by flight control routines for termination and rescheduling. FREQEJ should be called to complete the execution of any scheduled flight control routine or any nonscheduled routine which completes the execution of a scheduled one. FREQEJ reschedules the currently scheduled (and executing) waitlist routine using information from the appropriate Preprocessor data set. After the routine has been rescheduled, FREQEJ calls ENDJOB to return control to the simulator control loop.

FREQUP

Subroutine FREQUP performs the same function as FREQEJ, except that control is returned to the calling routine. For both FREQEJ and FREQUP, the rescheduling of a routine is skipped if the simulation time has reached the Preprocessor data set stoptime.

HDLNS

Subroutine HDLNS (HDLTIM, NDATA, MESAGE)

HDLNS may be called during a simulation to print messages and data and to store the messages and the data for a summary reprint at the end of the simulation. HDLTIM is a time to be printed with MESAGE. MESAGE is assumed to be a 8(6H) print line. NDATA specifies the number of COMMON block SCRTCH locations (0 to 10) to be printed on the line following the message. HDLNS also writes the message and the data onto unit 11. However, if NDATA is negative, no data is expected, and MESAGE is written onto unit 11 without being printed.

HEDLNP

HEDLNP is an entry point in subroutine HDLNS. HEDLNP is called by TERMIN for the reprint of all data and messages at the end of a simulation.

INOUT

Subroutine INOUT (ARRAY, ISTART, ISTOP)

INOUT is an entry point in subroutine INOUT and prints the variables ARRAY (ISTART) through ARRAY (ISTOP). These variables are printed using a floating-point format, five words to a line, omitting lines where all variables are zero.

INPUT

INPUT is a subroutine called by EXECT and RECYCL to read free-field simulator input cards. INPUT loads any of the COMMON blocks whose names are recognized by SOAP. INPUT may be called by the flight control, but all possible simulator initialization should be accomplished with the EXECT or RECYCL calls.

INPUT2

INPUT2 is an entry point in subroutine INPUT and is called by EXECT and RECYCL to read free-field simulator input cards for restart and recycle simulations, respectively.

ISPPRT

Subroutine ISPPRT is called by several Executive routines for diagnostic prints. It prints the current top wait list entry, the current flight time, and the elapsed computer time since the simulation started. This routine is primarily used for checkout runs to verify proper flight control sequencing.

JSPPRT

JSPPRT is an entry point in subroutine ISPPRT. JSPPRT is similar to ISPPRT except that the Executive routines call it prior to an Environment update. Instead of the wait list entry being printed, the mnemonic CALENV is printed to identify the Environment update.

KALNOW

KALNOW is a scheduled subroutine which sets the immediate update flag, LECCOM(704), to 1, forcing a call to the Environment. It is scheduled by CMDFIL.

MAIN

MAIN is the main program of the SOAP simulation. The simulation is initialized by MAIN calling EXECT. The simulation is started by calling EXECTR. Once EXECTR is called, control never returns to MAIN.

NEXTIM

Subroutine NEXTIM (T, DT, PDT, PC, TREF, TNEXT)

NEXTIM is used by the system scheduling logic to calculate the next time (TNEXT) a task is to be executed. The technique used is to multiply the delta time step (DT) by a pass counter (PC). The pass counter for a task is incremented by 1.0 each time NEXTIM is called. If the time step has changed then the previous time step (PDT) and the time of that change (TREF) are saved. At the time of the change, the pass counter (PC) is reset to 0.0. TNEXT is then computed as above and the TREF is included.

NINOUT

Subroutine NINOUT (ARRAY, ISTART, ISTOP)

NINOUT prints the variables ARRAY (ISTART) through ARRAY (ISTOP) much like INOUT. However, NINOUT uses a variable format for the print instead of the standard floating point of INOUT. For each variable, NINOUT determines the type of format to use. A floating-point format is used if the data is floating point, and integer if it is integer. Otherwise, a variable octal format is used with the Hollerith representation printed with the octal number. The octal format is the minimum format required to print the number. As in INOUT, the variables are printed five to a line with lines consisting of all zeroes being omitted.

PINOUT

Subroutine PINOUT (ARRAY, ISTART, ISTOP, IBLOCK)

PINOUT is another print output routine. PINOUT first prints the array identifier IBLOCK, a six-character or less BCD name. INOUT is then called to print ARRAY (ISTART) through ARRAY (ISTOP).

PLTOUT

Subroutine PLTOUT (N, IBCD, VAL, IFOR, LU, IPASS)

PLTOUT is used by the preprocessor-generated plot routines to dump data for subsequent plotting and may also be called by the flight software for the same purpose. N is the number of variables to be dumped and LU is the logical unit onto which the data is to be dumped (default is 20). IBCD, VAL, and IFOR are three arrays of length N. VAL contains the data to be plotted; IBCD the BCD (six characters or less) identifiers for the data; and IFOR the codes for the format of the data which will be dumped. The IFOR codes are one for floating point, two for integer, and three for octal. Prior to dumping the data, PLTOUT converts integer and octal data to floating point. If a flight software routine uses PLTOUT, it should set IPASS to a one for the first pass. PLTOUT will then dump the IBCD array onto LU for later use by the Plot Processor and reset IPASS to a zero. An additional feature for PLOTR is that an Environment update will be made prior to the plot dump if a simulator switch is set (LECCOM (579)=1). For the PLOTR1 through PLOTR9 routines, LECCOM (621) through LECCOM (629) contain the output logical units (default is 20) and LECCOM (611) through LECCOM (619) contain the Environment update switches.

PNNOUT

Subroutine PNNOUT (ARRAY, ISTART, ISTOP, IBLOCK)

PNNOUT is an entry point in subroutine PINOUT and performs the same function as PINOUT, except subroutine NINOUT is used to print the variables.

PRTSDT

PRTSDT is an entry point in subroutine SYSPRT and is called to print the schedule data tables, which consist of routine name, start time, stop time, delta time, priority, time of last delta time change, and schedule pass counter. PRTSDT can be scheduled.

PRWAIT

PRWAIT is an entry point in subroutine SYSPRT and is called to print the waitlist, which consists of scheduled routine's position on list (ELEMENT), time of next scheduled execution (CTABLE), name of routine scheduled (JOBTAB), priority (PRIOR), and location of job in scheduled data table. PRWAIT can be scheduled.

QZKILL

QZKILL is an entry point in subroutine RECOV. QZKILL is called by the UNIVAC 1110 system routine NERR$ whenever a fatal system error occurs (except for print limit or time limit). QZKILL calls RESTRT for either run termination or a RECYCL for another trajectory.

RECOV

Subroutine RECOV is written in assembly language and is called by EXECT at the start of a simulation. RECOV calls UNIVAC 1110 systems routines to set up the call to QZKILL in case a fatal system error occurs. RECOV is also called by RESTRT if the user has specified another trajectory to be run.

RECYCL

Subroutine RECYCL may be scheduled or called by the flight control routines to read additional simulator inputs at some time during a simulation and/or to recycle the simulation back to a previous State Image. Initially, RECYCL reads a data card which defines the type of action it is to perform. In one case, RECYCL calls INPUT to read additional data cards and then resumes the simulation. The other two cases involve a simulator restart at some previously dumped State Image. The difference between the latter two cases is that in one, new State Images are written over the old ones after the restart, while in the second case, a new simulator State Image file is begun. The former case would be used for recycling to an earlier State Image to correct an error, while the latter case represents a series of test cases run from one nominal starting point. In either case, RECYCL calls INPUT for additional simulator input after the desired State Image is obtained. For all three

types of RECYCLE action, optional State Image dumps are made prior to reading the first input card and after INPUT has been executed. Also, RECYCL calls FREQEJ to be rescheduled if appropriate.

RESTRT

Subroutine RESTRT is called by QZKILL whenever a fatal UNIVAC 1110 EXEC 8 error occurs. Normally RESTRT calls TERMIN to terminate the simulation properly. However, if LECCOM (580) is set to a 1, RESTRT calls RECOV to reestablish the termination linkages and then calls RECYCL for another trajectory.

REWTLT

Subroutine REWTLT (JOB, T, ISWITC, NENTRY)

REWTLT provides the flight control with the capability of altering and/or obtaining information from the waitlist, depending upon the setting of ISWITC. JOB is a six-character or less BCD name of the flight control routine to be considered. If ISWITC = 1, the presently scheduled waitlist time for JOB is replaced with T. If JOB does not appear on the waitlist, it is scheduled to occur at T. If ISWITC = 2, JOB is removed from the waitlist. If ISWITC = 3, the presently scheduled time of JOB is returned to T, and the waitlist is not altered. If JOB does not appear on the waitlist, T is set to -1.0. On every call to REWTLT, NENTRY is returned to the caller as the present position of JOB on the waitlist or set to a -1 if JOB does not appear on the waitlist. All three options are applied to the first occurrence of JOB in the waitlist.

SANDF

Subroutine SANDF (IFLAG, PRIOR, UPDT, TSTOP)

SANDF provides the Executive and flight control with a means of obtaining scheduling information about the flight control routine currently being executed. PRIOR is the routine's priority, UPDT the update frequency, and TSTOP the routine's stoptime. If IFLAG equals 0, it is set to a 1 by SANDF to indicate completion of the call and thus may be used as a first pass flag

by the caller. If IFLAG is not equal to 0, the location of the current job's buildit data in the buildit data table is returned.

SCHED

Subroutine SCHED (TSCHED, JOB, PRIOR, ISWITC)

SCHED is called by several Executive routines and may be called by flight control routines to perform one of three functions on the waitlist. The function to be performed is defined by ISWITC. If ISWITC = 0, the top flight control routine on the waitlist is removed. The preceding SCHED call is made by CONTRL during the simulation control loop if the current JOB does not reschedule itself. If ISWITC = 1, JOB (six-character BCD routine name) is scheduled (included on the waitlist) with priority PRIOR to be executed TSCHED seconds (flight time) in the future. If ISWITC = 2, JOB is scheduled with priority PRIOR to be executed at TSCHED. The last two SCHED functions are intended for use by the flight control

SELECT

SELECT is a FORTRAN subroutine written by the Preprocessor for use in the subsequent simulation. SELECT is called by CONTRL for the scheduled flight software execution in the simulator control loop. Entry points in SELECT represent the print routines and/or one of the plot routines PLOTR, PLOT1, ..., PLOTR8, PLOTR9 requested by Preprocessor input. These entry points may be called by the flight control although the entry point SELECT must never be called except by CONTRL. Also, Entry point MAXMIN may be represented in SELECT. The entry point MMPRNT for printing the maximum/minimum values is an entry point in select.

SIDUMP

Subroutine SIDUMP may be called by the flight control or scheduled to dump State Images onto tape or mass storage for later editing or restarts. SIDUMP is called by EXECT, RECYCL, and TERMIN. If scheduled, SIDUMP calls FREQEJ to be rescheduled and for termination.

C-2

SIPRIN

Subroutine SIPRIN may be called by the flight control or scheduled to print State Images for diagnostics. SIPRIN is called by EXECT, RECYCL, and TERMIN and uses PNNOUT to accomplish the print. If scheduled, SIPRIN calls FREQEJ to be rescheduled and for termination. SIPRIN contains 48 entry points which may be scheduled or called for individual COMMON block prints.

SORT3V

Subroutine SORT3V (ARRAY1, ARRAY2, ARRAY3, NVALUS)

SORT3V is used by SCHED to order the waitlist but may be called by a flight control routine. SORT3V sequences the first NVALUS locations of the three arrays ARRAY1, ARRAY2, and ARRAY3 in ascending order of the values in ARRAY1.

SREAD

Subroutine SREAD (LU)

SREAD is used by the Executive to read in the State Image dumps from a logical unit (LU), which is normally a 7 for restart runs and an 8 for recycle runs.

SUPTIM

Subroutine SUPTIM (ICAU, ICCER, IO)

SUPTIME computes the CAU, CC/ER, and I/O standard units of processing (SUP) charges accrued for the total runstream time to the time of the call. The routine is UNIVAC-dependent and is the basic SOAP timing algorithm. The basic time increment is 200 microseconds.

SWRITE

Subroutine SWRITE (LU)

SWRITE is used by the executive to write out the State Image dumps to a an LU, normally logical unit 8.

SYSPRT

SYSPRT is a SOAP executive system print routine, which can be scheduled or called. SYSPRT will print the schedule data tables and the current waitlist. These two prints are also accessible via entry points PRTSDT and PRWAIT, respectively. See these entry points for description of contents of the schedule data table and the waitlist prints.

TERMIN

TERMIN is the simulator termination subroutine and may be scheduled or called for termination of the simulation. TERMIN should be called for all abnormal simulator terminations as it terminates plot and State Image tapes which may then be used for diagnostic purposes. TERMIN first calls MMPRNT to print the maximum/minimum summary if MAXMIN was scheduled. TERMIN calls HEDLNP, an entry point in HDLNS, for a reprint of any "headlines" printed during the simulation. TERMIN calls SIDUMP for a final optional State Image dump, SYSPRT for a final scheduling summary and SIPRIN for a final diagnostic State Image print. TERMIN then puts end-of-files on the plot and State Image units and terminates the simulation.

TIMER

Subroutine TIMER (CALLER, ISWTCH, IEND)

TIMER is called to obtain a measure of the real time consumed by a particular routine during a simulation run. As many as 30 routines may be timed on the same run. The calling routine must supply the three calling arguments. CALLER is the name of the calling routine (Hollerith). ISWTCH is set to integer 1 to turn TIMER on, and then back to 0 to turn TIMER off.

In the most simple case, the calling routine would call TIMER at the beginning of its execution with ISWTCH = 1, and then at the end of its execution with ISWTCH = 0. However, if during its execution the routine to be timed passes control to some other routine, it may be desirable to turn TIMER off during this time and then back on when the routine being timed resumes execution.

3.3 SOAP SYSTEM COMMON BLOCKS

The SOAP executive recognizes the following COMMON block names:

ACCELC	BENDC	DPFSC1	GRADC	LNDGRC	RCS2C
ACSC	CONST	ENVCOM	GRAVC	MASPRC	RENDEC
ACTVEC	DATBUS	FSCOM	GUSTC	MASP1C	RMC
ACTV1C	DAVEHC	FSC1	GYRO1C	MASP2C	RNGNAC
ACTV2C	DAVE1C	FSC2	GYRO2C	NAVAIC	SCRTCH
AEROC	DCONST	FSC3	HORIZC	ORBALC	SLOSHC
AER1C	DGRAVC	FSC4	HORZ1C	ORBPAC	SNSCTS
AER2C	DIMUC	FSC5	HORZ2C	PASVEC	STARTC
AIRDAC	DMASSC	FS1C	IMUC	RADALC	THRUSC
ATMOSC	DPDBC	FS2C	LECCOM	RCSC	TVVCC
BARALC	DPFSC	GGTORC	LNDAIC	RCS1C	WINDC

Most of these blocks are associated with Environment models and the definitions of the variables stored therein are to be found in the documentation for the Environment models. These definitions can change from one version of the model to another. The COMMON blocks whose names start with the letters FS are associated with flight control; the definitions of the locations in these blocks are documented in section 5.

The SOAP system maintains the blocks CONST, DCONST, ENVCOM, LECCOM, and SCRTCH. The block SCRTCH is used for temporary storage during execution. No fixed locations are defined. Maps of the contents of the other four COMMON system blocks follow.

3.3.1 CONST COMMON BLOCK LENGTH = 48 PAGE 1 OF 2

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
1	WE	1	RAD/SEC	EARTH ANGULAR VELOCITY = 7.292115147E-5	
2	EGC	1	M**3/S**2	EARTH GRAVITATIONAL CONSTANT = 3.986011999995E+14	
3	PI	1	-	PI = 3.141592654	
4	PI2	1	-	2 * PI = 6.283185307	
5	PID2	1	-	PI / 2 = 1.570796327	
6	PDCDEL	1	-	SMALL NUMBER USED TO CHECK FOR ROUND-OFF = 0.0001	
7	AMIDNT	9	-	IDENTITY MATRIX = 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0	
16	FINITY	1	-	LARGE NUMBER = 1.0E+10	
17	SREGC	1	-	SQUARE ROOT OF EGC = 1.99649993738E+07	
18	FMUS	1	M**3/S**2	SUN GRAVITATIONAL CONSTANT = 1.327124989999E+20	
19	FMUM	1	M**3/S**2	MOON GRAVITATIONAL CONSTANT = 4.902779999997E+12	
20	AF	1	M	SEMI-MAJOR AXIS OF FISCHER ELLIPSOID = 6.378166E+06	
21	BF	1	M	SEMI-MINOR AXIS OF FISCHER ELLIPSOID = 6.356784284E+06	
22	RE	1	M	MEAN EQUITORIAL RADIUS OF THE EARTH = 6.378160E+06	
23	SLIGHT	1	M/S	SPEED OF LIGHT = 2.997925E+08	
24	FLAT	1	-	ELLIPTICITY OF FISCHER ELLIPSOID (1/298.3) = 0.003352329869	
25	ECC2	1	-	ECCENTRICITY OF THE EARTH SQUARED = 0.006693421623	
26	ENR	1	M	EARTH NORMAL RADIUS = .6355E+7	
27	AF2	1	M**2	SEMI-MAJOR AXIS OF FISCHER ELLIPSOID SQUARED = .4068100152E+14	
28	BF2	1	M**2	SEMI-MINOR AXIS OF FISCHER ELLIPSOID SQUARED = .4040870643E+14	

LOC	VARNAM	CONST DIM	UNITS	DEFINITION AND INITIAL VALUE
29	ITM	1	M/IN	CONVERT INCHES TO METERS = 2.540E-02
30	MTI	1	IN/M	CONVERT METERS TO INCHES = 3.937007874E+01
31	DTR	1	DEG/RAD	CONVERT DEGREES TO RADIANS = 1.745329252E-02
32	RTD	1	RAD/DEG	CONVERT RADIANS TO DEGREES = 5.729577951E+01
33	FTM	1	M/FT	CONVERT FEET TO METERS = 3.048E-01
34	MTF	1	FT/M	CONVERT METERS TO FEET = 3.280839895E+00
35	PTK	1	KG/LB	CONVERT POUNDS TO KILOGRAMS = 4.53592370E-01
36	KTP	1	LB/KG	CONVERT KILOGRAMS TO POUNDS = 2.204622622E+00
37	PTN	1	NT/LBF	CONVERT POUNDS FORCE TO NEWTONS = 4.448221615E+00
38	NTP	1	LBF/NT	CONVERT NEWTONS TO POUNDS FORCE = 2.248089431E-01
39	STK	1	KG/SLUG	CONVERT SLUGS TO KILOGRAMS = 1.459390294E+01
40	KTS	1	SLUG/KG	CONVERT KILOGRAMS TO SLUGS = 6.852176586E-02
41	FTN	1	NT-M/FT-LB	CONVERT FOOT-POUNDS TO NEWTON-METERS = 1.355817948E+00
42	NTF	1	FT-LB/NT-M	CONVERT NEWTON-METERS TO FOOT-POUNDS = 7.375621493E-01
43	PTS	1	SLUG/LB	CONVERT POUNDS TO SLUGS = 3.108095017E-02
44	STP	1	LB/SLUG	CONVERT SLUGS TO POUNDS = 3.217404856E+01
45	IME	1	SF2/KM2	CONVERT KG-M**2 TO SLUG-FT**2 = 7.375621493E-01
46	IEM	1	KM2/SF2	CONVERT SLUG-FT**2 TO KG-M**2 = 1.355817948E+00
47	PME	1	P/F2 / N/M2	CONVERT NT/M**2 TO LB/FT**2 = 2.088543423E-02
48	PFM	1	N/M2 / P/F2	CONVERT LB/FT**2 TO NT/M**2 = 4.788025898E+01

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE
1	DPWE	1	RAD/SEC	EARTH ANGULAR VELOCITY = .7292115146668192000D-04
2	DPEGC	1	M**3/S**2	EARTH GRAVITATIONAL CONSTANT = 3.986011999950+14
3	DPPI	1	-	PI = .3141592653589793240+01
4	DPPI2	1	-	2 * PI = .6283185307179586470+01
5	DPPID2	1	-	PI / 2 =.1570796326794896620+01
6	DPROCD	1	-	SMALL NUMBER USED TO CHECK FOR ROUND-OFF = 0.100-03
7	DPAMID	9	-	IDENTITY MATRIX = 1.0D 0.0D 0.0D 0.0D 1.0D 0.0D 0.0D 0.0D 1.0D
16	DFNITY	1	-	LARGE NUMBER = 1.0D+10
17	DSREGC	1	-	SQUARE ROOT OF EGC = 1.996499937380+07
18	DPFMUS	1	M**3/S**2	SUN GRAVITATIONAL CONSTANT = 1.327124989990+20
19	DPFMUM	1	M**3/S**2	MOON GRAVITATIONAL CONSTANT = 4.902779999970+12
20	DPAF	1	M	SEMI-MAJOR AXIS OF FISCHER ELLIPSOID = 6.3781660+06
21	DPBF	1	M	SEMI-MINOR AXIS OF FISCHER ELLIPSOID = 6.356784283607106930+06
22	DPRE	1	M	MEAN EQUITORIAL RADIUS OF THE EARTH = 6.3781600+06
23	DPLITE	1	M/S	SPEED OF LIGHT = 2.9979250+08
24	DPFLAT	1	-	ELLIPTICITY OF FISCHER ELLIPSOID (1/298.3) = 0.335232986925913510D-02
25	DPECC2	1	-	ECCENTRICITY OF THE EARTH SQUARED = 0.669342162296594470-02
26	DPENR	1	M	EARTH NORMAL RADIUS = .63550+7
27	DPAF2	1	M**2	SEMI-MAJOR AXIS OF FISCHER ELLIPSOID SQUARED = .406810015235560D+14
28	DPBF2	1	M**2	SEMI-MINOR AXIS OF FISCHER ELLIPSOID SQUARED = .4040870642831431970+14

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
29	DITM	1	M/IN	CONVERT INCHES TO METERS = 2.540D-02	
30	DMTI	1	IN/M	CONVERT METERS TO INCHES = 3.937007874015748D+01	
31	DDTR	1	DEG/RAD	CONVERT DEGREES TO RADIANS = 1.745329251994329576D-02	
32	DRTD	1	RAD/DEG	CONVERT RADIANS TO DEGREES = 5.729577951308232088D+01	
33	DFTM	1	M/FT	CONVERT FEET TO METERS = 3.048D-01	
34	DMTF	1	FT/M	CONVERT METERS TO FEET = 3.280839895013123360D+00	
35	DPTK	1	KG/LB	CONVERT POUNDS TO KILOGRAMS = 4.53592370D-01	
36	DKTP	1	LB/KG	CONVERT KILOGRAMS TO POUNDS = 2.204622621848775800+00	
37	DPTN	1	NT/LBF	CONVERT POUNDS FORCE TO NEWTONS = 4.448221615260500D+00	
38	DNTP	1	LBF/NT	CONVERT NEWTONS TO POUNDS FORCE = 2.248089430997104830-01	
39	DSTK	1	KG/SLUG	CONVERT SLUGS TO KILOGRAMS = 1.459390293720636480+01	
40	DKTS	1	SLUG/KG	CONVERT KILOGRAMS TO SLUGS = 6.852176585679175530-02	
41	DFTN	1	NT-M/FT-LB	CONVERT FOOT-PUNDS TO NEWTON-METERS = 1.355817948331400390+00	
42	DNTF	1	FT-LB/NT-M	CONVERT NEWTON-METERS TO FOOT-POUNDS = 7.375621492772653660-01	
43	DPTS	1	SLUG/LB	CONVERT POUNDS TO SLUGS = 3.108095017156725280-02	
44	DSTP	1	LB/SLUG	CONVERT SLUGS TO POUNDS = 3.217404855643044610+01	
45	DIME	1	SF2/KM2	CONVERT KG-M**2 TO SLUG-FT**2 = 7.375621492772653660-01	
46	DIEM	1	KM2/SF2	CONVERT SLUG-FT**2 TO KG-M**2 = 1.355817948331400390+00	
47	DPME	1	P/F2 / N/M2	CONVERT NT/M**2 TO LB/FT**2 = 2.088543423315012690-02	
48	DPEM	1	N/M2 / P/F2	CONVERT LB/FT**2 TO NT/M**2 = 4.788025898033584260D+01	

3.3.3 ENVCOM COMMON BLOCK LENGTH = 1000 PAGE 1 OF 3

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
1	------	120	-	UNUSED LOCATIONS	
121	CMDF	40	-	DEFINED BY MODELS AND REPRESENTS THE ID OF THE FIRST COMMAND IN A COMMAND SET.	
161	CMDL	40	-	DEFINED BY MODELS AND REPRESENTS THE ID OF THE LAST COMMAND IN A COMMAND SET.	
201	IJK	40	-	A THREE BIT FLAGWORD (INTEGER 0 TO 7) ASSOCIATED WITH A COMMAND SET.I=1 MEANS THAT DYNAMIC SENSOR RECIPES ARE INVALIDATED WHEN A COMMAND FROM THE SET IS ISSUED.J=1 IS FOR NON-DYNAMIC SENSOR RECIPES.K=1 MEANS THAT AN IMMEDIATE ENVIRONMENT UPDATE IS REQUIRED WHENEVER A COMMAND FROM THE SET IS ISSUED.	
241	TONDEL	40	SEC	ON DELAY FOR COMMANDS IN THE CORRESPONDING COMMAND SET.	
281	MODEL	40	-	NAME OF THE MODEL WHICH ACCEPTS THE CORRESPONDING COMMAND SET.	
321	NSETS	1	-	NUMBER OF COMMAND SETS.	
322	INIT	1	-	ENVIRONMENT INITIALIZATION FLAG, =1 FOR FIRST PASS INITIALIZATION,=2 FOR SECOND, AND =0 FOR NORMAL UPDATE CALL.	
323	NCMMDS	1	-	SET TO THE NUMBER OF COMMANDS AT THE TOP OF THE COMMAND LIST THAT A MODEL IS TO EXECUTE.	
324	NUMMOD	1	-	TOTAL NUMBER OF ENVIRONMENT MODELS, CURRENTLY SET TO 32.	
325	NDYMOD	1	-	TOTAL NUMBER OF DYNAMICS ASSOCIATED MODELS,CURRENTLY SET TO 14.	
326	NDYSET	1	-	NUMBER OF DYNAMICS COMMAND SETS.	
327	T	1	SEC	CURRENT ACTIVE VEHICLE TIME.	
328	TSTEP	1	SEC	TIME TO WHICH THE ACTIVE VEHICLE IS TO UPDATE.	
329	NMAX	1	-	MAXIMUM NUMBER OF COMMANDS WHICH CAN BE ACCUMULATED,CURRENTLY SET TO 49.	
330	NPUSH	1	-	CURRENT NUMBER OF COMMANDS ON THE PUSH LIST.	

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
331	NAME	50	-	BCD GENERAL MODEL NAMES.	
381	LSW	50	-	MODEL PRESENT SWITCHES SET BY MODELS TO A ONE IF THEY ARE PRESENT.	
431	TPUSH	50	-	THE TIMES AT WHICH COMMANDS ON THE COMMAND PUSH LIST ARE TO BE EXECUTED.	
481	CMMD	50	-	PART OF THE COMMAND PUSH LIST CONTAINING THE IDS OF THE COMMANDS TO BE EXECUTED.	
531	DAT1	50	-	PART OF THE COMMAND PUSH LIST CONTAINING THE FIRST PIECE OF DATA FOR THE COMMAND.	
581	DAT2	50	-	PART OF THE COMMAND PUSH LIST CONTAINING THE SECOND PIECE OF DATA FOR THE COMMAND.	
631	DAT3	50	-	PART OF THE COMMAND PUSH LIST CONTAINING THE THIRD PIECE OF DATA FOR THE COMMAND.	
681	TOFDEL	40	SEC	OFF DELAY FOR COMMANDS IN THE CORRESPONDING COMMAND SET.	
721	MODELN	40	-	NUMBER IDENTIFYING MODEL ASSOCIATED WITH CORRESPONDING COMMAND SET.	
761	NMODEL	50	-	PART OF THE COMMAND PUSH LIST CONTAINING A NUMBER IDENTIFYING THE MODEL ASSOCIATED WITH A COMMAND.	
811	STEPDT	50	SEC	MAXIMUM MODEL UPDATE INTERVAL WHICH MAY BE DEFINED FOR USE BY ACTIVE VEHICLE OR SENSOR CONTROL.	
861	AVDDT	1	SEC	ACTIVE VEHICLE UPDATE INTERVAL.	
862	SCDDT	1	SEC	SENSOR CONTROL UPDATE INTERVAL.	
863	TSC	1	SEC	CURRENT SENSOR CONTROL TIME.	
864	IDYN	1	-	FLAG WHICH SPECIFIES ACTIVE VEHICLES ACTION ON A GIVEN CALL. =1,RECALCULATE DYNAMIC DERIVITIVES BEFORE STARTING UPDATE. =0,START UPDATE WITHOUT RECALCULATING DERIVITIVES. =-1,UPDATE DERIVITIVES ONLY,DONT ADVANCE TIME.	

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
865	ICMMD	1	-	FLAG INFORMING ACTIVE VEHICLE OF WHAT IS GOING TO HAPPEN AFTER UPDATE TO TSTEP. = 1,DYNAMICS COMMANDS TO BE EXECUTED AT TSTEP. = 0,NO COMMANDS AT TSTEP. =-1,DYNAMICS AND NONDYNAMICS COMMANDS TO BE EXECUTED AT TSTEP.	
866	AVTNEW	1	SEC	CURRENT ACTIVE VEHICLE TIME.	
867	SCTNEW	1	SEC	CURRENT SENSOR CONTROL TIME.	
868	IRKAV	1	-	RUNGA-KUTTA PASS FLAG FOR ACTIVE VEHICLE.	
869	EPOCH	1	-	TIME IN DAYS FROM JULY 1 TO START OF RUN	
870	IRUIN	50	-	FLAG TABLE ASSOCIATED WITH COMMAND PUSH LIST DEFINING COMMANDS EFFECT ON DYNAMIC RECIPES. =1,COMMAND INVALIDATES RECIPES	
920	JRUIN	50	-	NONDYNAMIC COUNTERPART TO IRUIN.	
970	CEPOCH	1	SEC	EPOCH PLUS CURRENT TIME	
971	ITYPEM	20	-	MODEL TYPE INDICATOR FOR COMMANDS TO ENVIRONMENT DYNAMIC MODELS. =0 INDICATES THE MODEL HAS AN INSTANTANEOUS (NO LAGS) RESPONSE TO A COMMAND. =1 INDICATES THE MODEL HAS A LAG RESPONSE TO A COMMAND, THAT IS, THE MODEL DYNAMICS DOES NOT CHANGE UNLESS THERE IS A TIME INCREMENT.	
991	ITYPEC	1	-	INDICATES TYPE COMMAND ISSUED TO MODELS	
992	PTCALL	1	SEC	PREVIOUS TIME TO WHICH THE ENVIRONMENT HAS UPDATED TO.	
993	TAVREQ	1	SEC	TIME OF ACTIVE VEHICLE NEXT REQUESTED UPDATE	
994	TSCREQ	1	SEC	TIME OF SENSOR CONTROL NEXT REQUESTED UPDATE	
995	------	6	-	UNUSED LOCATIONS	

3.3.4 LECCOM COMMON BLOCK-VERSION 805D2 LENGTH = 1500 PAGE 1

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
	TIME	1	SEC	CURRENT FLIGHT SOFTWARE TIME.	
2	JOB	1	—	BCD NAME OF THE CURRENTLY SCHEDULED ROUTI...	
3	NJOB	1	—	NUMBER OF ROUTINE SCHEDULE CARDS READ BY THE PREPROCESSOR.	
4	CTABLE	50	SEC	PART OF THE WAITLIST, CONTAINING THE SIMULATION TIMES AT WHICH THE FLIGHT SOFTWARE ROUTINES OF JOBTAB ARE TO BE EXECUTED.	
54	JOBTAB	50	—	PART OF THE WAITLIST, CONTAINING THE NAMES OF THE FLIGHT SOFTWARE ROUTINES SCHEDULED TO BE EXECUTED.	
104	PRIOR	50	—	PART OF THE WAITLIST, CONTAINING THE PRIORITIES OF THE JOBTAB ROUTINES.	
154	LIST	1	—	NUMBER OF WAITLIST ENTRIES.	
155	SUBNAM	100	—	PREPROCESSOR SCHEDULE CARD ROUTINE NAMES.	
255	APRI	100	—	PREPROCESSOR SCHEDULE CARD ROUTINE PRIORITIES.	
355	AUPDT	100	SEC	PREPROCESSOR SCHEDULE CARD ROUTINE UPDATE FREQUENCIES.	
455	STT	100	SEC	PREPROCESSOR SCHEDULE CARD ROUTINE STOP TIMES.	
555	NDPCB	1	—	NUMBER OF DOUBLE PRECISION COMMON BLOCKS	
556	IDPCBN	10	—	DOUBLE PRECISION COMMON BLOCK NAMES	
572	------	13	—	UNUSED LOCATIONS	
579	IPSW	1	—	INITIALLY ZERO, IF NONZERO, AN ENVIRONMENT UPDATE WILL BE MADE WHENEVER THE PREPROCESSOR GENERATED ROUTINE PLOTR IS CALLED FOR A PLOT DUMP.	
580	IRECY	1	—	NORMALLY ZERO, A NONZERO VALUE WILL RESULT IN A RECYCLE FOR ANOTHER TRAJECTORY IF A SIMULATION ENCOUNTERS A FATAL 1108 ERROR. NORMALLY THE SIMULATION IS TERMINATED VIA TERMI...	
581	TINIT	1	SEC	INITIAL CAU SUP TIME IN SECONDS.	
582	XTIM	1	SEC	ELAPSED REAL TIME SINCE TINIT (MAINTAINED ONLY IF ISP NOT EQUAL TO ZERO.	

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
582	I01	1	200 M/S SUPS	INITIAL I/O SUP TIME (1 UNIT = APPX. 200 MICROSECONDS).	
584	TI01	1	SEC	INITIAL I/O SUP TIME IN SECONDS.	
585	IENVUP	1	-	INDICATES ENVIRONMENT UPDATE IS IN PROGRESS. =1 UPDATE IN PROGRESS =0 ENV NOT CURRENTLY BEING UPDATED	
586	M	1	-	A NONZERO M RESULTS IN PLTOUT DUMPING THE PLOTR SYMBOL RECORD.	
587	IABORT	1	SEC	KILLS A LOOP IN TERMIN IN CASE OF AN ENVIRONMENT ABORT	
588	ISP	1	-	INITIALLY ZERO, SSFS SYSTEM DIAGNOSTIC PRINT SWITCH. =1, EACH TIME A SCHEDULED ROUTINE IS EXECUTED, THE ROUTINE NAME, THE CURRENT FLIGHT SOFTWARE TIME, AND THE ELAPSED REAL TIME SINCE INITIALIZATION ARE PRINTED. FOR EACH ENVIRONMENT CALL, THE MNEMONIC CALENV IS PRINTED INSTEAD OF THE ROUTINE NAME. =2, THE INFORMATION DEFINED ABOVE IS PRINTED AND THE CALLING ARGUMENTS OF THE SCHED ROUTINE ARE PRINTED EACH TIME IT IS CALLED. =3, THE INFORMATION DEFINED ABOVE IS PRINTED, THE WAITLIST IS PRINTED EACH TIME SCHED IS CALLED, AND THE CALLING ARGUMENTS OF REWTLT ARE PRINTED EACH TIME IT IS CALLED.	
589	------	9	-	UNUSED LOCATIONS	
592	LU29	1	-	LOGICAL UNIT 29 - SYSTEM SCRATCH	
593	LU20	1	-	LOGICAL UNIT 20 - DEFAULT PLOT FILE	
600	LU15	1	-	LOGICAL UNIT 15 - THERMAL PROTECT SYS	
601	LU11	1	-	LOGICAL UNIT 11 - HEADLINES PRINT SUMMARY	
602	LU10	1	-	LOGICAL UNIT 10 - AEROELASTICS DATA	
603	LU9	1	-	LOGICAL UNIT 9 - AERO DATA FILE	
604	LU8	1	-	LOGICAL UNIT 8 - SSFS STATE IMAGE OUTPUT	
605	LU7	1	-	LOGICAL UNIT 7 - SSFS STATE IMAGE INPUT	
606	LU4	1	-	LOGICAL UNIT 4 - MAP DIRECTIVES	
607	LU3	1	-	LOGICAL UNIT 3 - SELECT, CALMOD, DUMMYS	
608	LOUT	1	-	LOGICAL UNIT 6 - FORTRAN WRITE FILE	
609	LIN	1	-	LOGICAL UNIT 5 - FORTRAN READ UNIT	
610	------	1	-	UNUSED LOCATION	

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
611	L611	1	—	ENVIRONMENT UPDATE FLAG FOR PLOTR1 INITIALIZED TO ZERO. IF SET TO A ONE, AN ENVIRONMENT UPDATE WILL BE MADE PRIOR TO THE PLOTR1 PLOT DUMP.	
612	L612	1	—	ENVIRONMENT UPDATE FLAG FOR PLOTR2.	
613	L613	1	—	ENVIRONMENT UPDATE FLAG FOR PLOTR3.	
614	L614	1	—	ENVIRONMENT UPDATE FLAG FOR PLOTR4.	
615	L615	1	—	ENVIRONMENT UPDATE FLAG FOR PLOTR5.	
616	L616	1	—	ENVIRONMENT UPDATE FLAG FOR PLOTR6.	
617	L617	1	—	ENVIRONMENT UPDATE FLAG FOR PLOTR7.	
618	L618	1	—	ENVIRONMENT UPDATE FLAG FOR PLOTR8.	
619	L619	1	—	ENVIRONMENT UPDATE FLAG FOR PLOTR9.	
620	------	1	—	UNUSED LOCATION.	
621	ILU1	1	—	PLOT DUMP UNIT FOR PLOTR1 = 20.	
622	ILU2	1	—	PLOT DUMP UNIT FOR PLOTR2 = 20.	
623	ILU3	1	—	PLOT DUMP UNIT FOR PLOTR3 = 20.	
624	ILU4	1	—	PLOT DUMP UNIT FOR PLOTR4 = 20.	
625	ILU5	1	—	PLOT DUMP UNIT FOR PLOTR5 = 20.	
626	ILU6	1	—	PLOT DUMP UNIT FOR PLOTR6 = 20.	
627	ILU7	1	—	PLOT DUMP UNIT FOR PLOTR7 = 20.	
628	ILU8	1	—	PLOT DUMP UNIT FOR PLOTR8 = 20.	
629	ILU9	1	—	PLOT DUMP UNIT FOR PLOTR9 = 20.	
630	------	1	—	UNUSED LOCATION.	
631	IFP1	1	—	FIRST PASS FLAG FOR PLOTR1.	
632	IFP2	1	—	FIRST PASS FLAG FOR PLOTR2.	
633	IFP3	1	—	FIRST PASS FLAG FOR PLOTR3.	
634	IFP4	1	—	FIRST PASS FLAG FOR PLOTR4.	
635	IFP5	1	—	FIRST PASS FLAG FOR PLOTR5.	
636	IFP6	1	—	FIRST PASS FLAG FOR PLOTR6.	
637	IFP7	1	—	FIRST PASS FLAG FOR PLOTR7.	
638	IFP8	1	—	FIRST PASS FLAG FOR PLOTR8.	
639	IFP9	1	—	FIRST PASS FLAG FOR PLOTR9.	
640	MMFPF	1	—	MAXMIN FIRST PASS FLAG	
641	DT	10	SEC	INITIALIZED TO -1.0, THESE LOCATIONS MAY BE INITIALIZED TO SPECIFY DELTA FLIGHT TIMES TO BE SKIPPED OVER BY ADVAN.	
651	------	50	—	UNUSED LOCATIONS	
701	NSPCB	1	—	NUMBER OF SINGLE PRECISION COMMON BLOCKS	
702	------	2	—	UNUSED LOCATIONS	
704	KALNOW	1	—	IF NONZERO, THE SYSTEM WILL UPDATE THE ENVIRONMENT BEFORE EXECUTING THE NEXT SCHEDULED ROUTINE AND THEN RESET THE FLAG.	
705	JOBLBD	50	—	JOB LOCATION IN BUILDIT DATA	
755	LJOBID	1	—	LAST JOB BUILDIT DATA ID	

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
756	TSTART	100	SEC	PREPROCESSOR SCHEDULE CARD ROUTINE START TIMES	
856	------	12	-	UNUSED LOCATIONS	
868	TCALL	1	SEC	TIME TO WHICH ENVIRONMENT IS TO UPDATE.	
869	REASON	1	-	ENVIRONMENT UPDATE FLAG, SET TO A ONE FOR A FULL UPDATE OR TO A ZERO FOR A DYNAMICS ONLY UPDATE.	
870	------	4	-	UNUSED LOCATIONS	
874	INITC	1	-	COMMUNICATOR FIRST PASS FLAG.	
875	ICOEF	1	-	RANDOM NUMBER GENERATOR CONSTANT.	
876	K235M1	1	-	RANDOM NUMBER GENERATOR CONSTANT.	
877	C235M1	1	-	RANDOM NUMBER GENERATOR CONSTANT.	
878	KOUNT	1	-	RANDOM NUMBER GENERATOR CONSTANT.	
879	NAME4	4	-	RANDOM NUMBER GENERATOR PRINT DATA.	
883	IFRMAT	18	-	RANDOM NUMBER GENERATOR PRINT DATA.	
901	IMAGNO	1	-	NUMBER OF LAST STATE IMAGE DUMP.	
902	NAIME	1	-	BCD NAME OF SIMULATION TYPE.	
903	IWHERE	1	-	SYSTEM FLAG USED ON RESTARTS OR RECYCLES.	
904	IRETN	1	-	SYSTEM FLAG FOR ENVIRONMENT UPDATES AT STATE IMAGE DUMPS.	
905	SIMTME	1	SEC	FLIGHT SOFTWARE TIME AT INITIALIZATION.	
906	ITIME1	1	200 M/S SUPS	INITIAL CAU SUP TIME (1 UNIT = APPX. 200 MICROSECONDS).	
907	INOUT	1	-	INITIALLY ZERO, THIS INTEGER NUMBER MAY BE USED TO DEFINE THE GENERATIVE NUMBER OF THE TWO RANDOM NUMBER GENERATORS. IF ZERO, THE SYSTEM WILL ARBITRARILY DEFINE A BASE.	
908	LUOUT	1	-	INITIALIZED AT 8, LUOUT DEFINES THE DUMP UNIT FOR STATE IMAGE DUMPS. IF SET TO A 7, THE INPUT AND OUTPUT STATE IMAGE UNITS ARE THE SAME. LUOUT SHOULD ONLY BE CHANGED ON AN ORIGINAL RUN, SINCE ON A RESTART RUN ONE STATE IMAGE DUMP IS MADE BEFORE THE INPUT DATA IS READ AND THUS BEFORE LUOUT IS CHANGED.	
909	IBASE	1	-	RANDOM NUMBER GENERATOR GENERATIVE NUMBER.	
910	INITR	1	-	RANDOM NUMBER GENERATOR INIT FLAG	
911	IRNPRT	1	-	RANDOM NUMBER PRINT FLAG =0 NO PRINT =1 PRINT ON (DEFAULT)	
912	ICCER	1	200 M/S SUPS	INITIAL CC/ER SUP TIME (1 UNIT = APPX. 200 MICROSECONDS).	
913	TCCER	1	SEC	INITIAL CC/ER SUP TIME IN SECONDS.	
914	------	20	-	UNUSED LOCATIONS.	

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
934	PME	1	(LBF/FT2) / (NT/M2)	CONVERSION FACTOR, NEWTONS PER METERS**2 TO POUNDS PER FEET**2 = 0.2088543423E-01	
935	PEM	1	(NT/M2) / (LBF/FT2)	CONVERSION FACTOR, POUNDS PER FEET**2 TO NEWTONS PER METERS**2 = 0.4788025898E+02	
936	ITM	1	M/IN	CONVERSION FACTOR, INCHES TO METERS = 0.0254	
937	DTR	1	RAD/DEG	CONVERSION FACTOR, DEGREES TO RADIANS = 0.01745329252	
938	RTD	1	DEG/RAD	CONVERSION FACTOR, RADIANS TO DEGREES = 57.29577951	
939	FTM	1	M/FT	CONVERSION FACTOR, FEET TO METERS=.3048	
940	MTF	1	FT/M	CONVERSION FACTOR, METERS TO FEET = 3.280839895	
941	PTK	1	KG/LB	CONVERSION FACTOR, POUNDS TO KILOGRAMS = 0.43059237	
942	KTP	1	LB/KG	CONVERSION FACTOR, KILOGRAMS TO POUNDS = 2.204622622	
943	PTN	1	NT/LBF	CONVERSION FACTOR, POUNDS TO NEWTONS = 4.448221615	
944	NTP	1	LBF/NT	CONVERSION FACTOR, NEWTONS TO POUNDS = 0.2248089431	
945	STK	1	KG/SLUGS	CONVERSION FACTOR, SLUGS TO KILOGRAMS = 14.59390294	
946	KTS	1	SLUGS/KG	CONVERSION FACTOR, KILOGRAMS TO SLUGS = 0.06852176586	
947	FTN	1	NT-M/FT-LBF	CONVERSION FACTOR, FOOT-POUNDS TO NEWTON-METERS = 1.355817948	
948	NTF	1	FT-LBF/NT-M	CONVERSION FACTOR, NEWTON-METERS TO FOOT-POUNDS = 0.7375621493	
949	PTS	1	SLUGS/LB	CONVERSION FACTOR, POUNDS TO SLUGS = 0.03108095017	
950	STP	1	LB/SLUGS	CONVERSION FACTOR, SLUGS TO POUNDS = 32.17404856	
951	IME	1	SF2/KM2	CONVERSION FACTOR, KILOGRAM-METERS**2 TO SLUG-FT**2 = 0.7375621493	
952	IEM	1	KM2/SF2	CONVERSION FACTOR, SLUG-FT**2 TO KILOGRAM-METERS**2 = 1.355817948	
953	IPASS	1	-	PASS COUNTER FOR BLKPLT ROUTINE.	
954	------	43	-	UNUSED LOCATION	
997	NCT	1	-	HEDLNS PASS COUNTER.	
998	ISIDMP	1	-	SWITCH CONTROLLING AUTOMATIC STATE IMAGE DUMPS, INITIALIZED TO ZERO FOR NO DUMPS. IF SET TO A ONE, STATE IMAGE DUMPS WILL OCCUR AFTER THE SIMULATION INITIALIZATION DATA IS READ, AFTER THE ENVIRONMENT INITIALIZATION, AT TERMINATION BY TERMIN,	

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
				BEFORE AND AFTER INITIALIZATION DATA IS READ ON NORMAL RECYCLES, AFTER THE INITIALIZATION DATA IS READ ON AN SI RECYCLE, BEFORE AND AFTER INITIALIZATION DATA IS READ ON AN SISAVE RECYCLE(BOTH ON THE NEW FILE).	
999	IPNLU	1	-	FIRST PASS FLAG FOR ALTERNATE PLOTR UNIT.	
1000	NLU	1	-	ALTERNATE PLOTR DUMP UNIT. RESET TO A ZERO AFTER EACH PLOTR CALL.	
1001	SPC	100	-	SCHEDULED ROUTINE PASS COUNTER	
1101	TLDTC	100	SEC	TIME OF LAST DT CHANGE FOR SCHEDULED ROUTINE.	
1201	------	100	-	UNUSED LOCATIONS	
1301	J1	1	-	MAXIMUM LENGTH OF ACTVEH COMMON BLOCK	
1302	J2	1	-	MAXIMUM LENGTH OF IMU COMMON BLOCK	
1303	J3	1	-	MAXIMUM LENGTH OF RCS COMMON BLOCK	
1304	J4	1	-	MAXIMUM LENGTH OF TVC COMMON BLOCK	
1305	J5	1	-	MAXIMUM LENGTH OF THRUST COMMON BLOCK	
1306	J6	1	-	MAXIMUM LENGTH OF ACS COMMON BLOCK	
1307	J7	1	-	MAXIMUM LENGTH OF ACCEL COMMON BLOCK	
1308	------	1	-	UNUSED LOCATION	
1309	J9	1	-	MAXIMUM LENGTH OF MASPRO COMMON BLOCK	
1310	J10	1	-	MAXIMUM LENGTH OF AERO COMMON BLOCK	
1311	J11	1	-	MAXIMUM LENGTH OF GRAV COMMON BLOCK	
1312	J12	1	-	MAXIMUM LENGTH OF ATMOS COMMON BLOCK	
1313	J13	1	-	MAXIMUM LENGTH OF SLOSH COMMON BLOCK	
1314	J14	1	-	MAXIMUM LENGTH OF BEND COMMON BLOCK	
1315	J15	1	-	MAXIMUM LENGTH OF GYRO COMMON BLOCK	
1316	J16	1	-	MAXIMUM LENGTH OF PASVEH COMMON BLOCK	
1317	J17	1	-	MAXIMUM LENGTH OF GUST COMMON BLOCK	
1318	J18	1	-	MAXIMUM LENGTH OF HORIZS COMMON BLOCK	
1319	J19	1	-	MAXIMUM LENGTH OF STARTR COMMON BLOCK	
1320	J20	1	-	MAXIMUM LENGTH OF RENDEZ COMMON BLOCK	
1321	J21	1	-	MAXIMUM LENGTH OF ORBALT COMMON BLOCK	
1322	------	3	-	UNUSED LOCATIONS	
1325	J25	1	-	MAXIMUM LENGTH OF GGTORK COMMON BLOCK	
1326	J26	1	-	MAXIMUM LENGTH OF ORBPAR COMMON BLOCK	
1327	------	5	-	UNUSED LOCATIONS	
1332	J32	1	-	MAXIMUM LENGTH OF SNSCTL COMMON BLOCK	
1333	J33	1	-	MAXIMUM LENGTH OF BDCOM COMMON BLOCK	
1334	J34	1	-	MAXIMUM LENGTH OF RM COMMON BLOCK	
1335	J35	1	-	MAXIMUM LENGTH OF AER1 COMMON BLOCK	
1336	J36	1	-	MAXIMUM LENGTH OF RCS1 COMMON BLOCK	
1337	J37	1	-	MAXIMUM LENGTH OF MASPR1 COMMON BLOCK	
1338	J38	1	-	MAXIMUM LENGTH OF ACTVE1 COMMON BLOCK	
1339	J39	1	-	MAXIMUM LENGTH OF GYRO1 COMMON BLOCK	

```
1340      J40    1    |  MAXIMUM LENGTH OF DAVEH COMMON BLOCK
1341      J41    1    |  MAXIMUM LENGTH OF DIMU COMMON BLOCK
1342      J42    1    |  MAXIMUM LENGTH OF DMASS COMMON BLOCK
1343      J43    1    |  MAXIMUM LENGTH OF DGRAV COMMON BLOCK
1344      J44    1    |  MAXIMUM LENGTH OF DAVE1 COMMON BLOCK
1345      J45    1    |  MAXIMUM LENGTH OF DPDB COMMON BLOCK
1346      J46    1    |  MAXIMUM LENGTH OF DFSC COMMON BLOCK
1347      J47    1    |  MAXIMUM LENGTH OF DFSC1 COMMON BLOCK
1348      J54    1    |  MAXIMUM LENGTH OF HORZ1 COMMON BLOCK
1349    -------  2    |  UNUSED LOCATIONS
1351      K1     1    |  MAXIMUM LENGTH OF LECCOM COMMON BLOCK
1352      K2     1    |  MAXIMUM LENGTH OF FSCOM COMMON BLOCK
1353      K3     1    |  MAXIMUM LENGTH OF ENVCOM COMMON BLOCK
1354      K4     1    |  MAXIMUM LENGTH OF F1 COMMON BLOCK
1355      K5     1    |  MAXIMUM LENGTH OF F2 COMMON BLOCK
1356      K6     1    |  MAXIMUM LENGTH OF F3 COMMON BLOCK
1357      K7     1    |  MAXIMUM LENGTH OF F4 COMMON BLOCK
1358      K8     1    |  MAXIMUM LENGTH OF F5 COMMON BLOCK
1359      K9     1    |  MAXIMUM LENGTH OF FS1 COMMON BLOCK
1360    -------  1    |  UNUSED LOCATION
1361      K10    1    |  MAXIMUM LENGTH OF FS2 COMMON BLOCK
1401      J49    1    |  MAXIMUM LENGTH OF AER2 COMMON BLOCK
1402      J50    1    |  MAXIMUM LENGTH OF RCS2 COMMON BLOCK
1403      J51    1    |  MAXIMUM LENGTH OF MASPR2 COMMON BLOCK
1404      J52    1    |  MAXIMUM LENGTH OF ACTVE2 COMMON BLOCK
1405      J53    1    |  MAXIMUM LENGTH OF GYRO2 COMMON BLOCK
1406      J55    1    |  MAXIMUM LENGTH OF HORZ2 COMMON BLOCK
1407    -------  24   |  UNUSED LOCATIONS
1431      IBLK   56   |  LIST OF ALL COMMON BLOCK NAMES
```

3.4 MATH, FLIGHT CONTROL, AND ENVIRONMENT UTILITY ROUTINES

These routines supplement the FORTRAN mathematical library and serve to support Environment and/or Flight Control routines. Typically, the user will call these routines rather than schedule them. In places where both single- and double-precision versions of the routines are available, one of the writeups is considerably more extensive. None of these routines has a dedicated COMMON block, although some use information stored in specified locations of other COMMON blocks.

Routines described in this section include:

ABVAL	DOT	MATYPR
ACCOSH	DOTDP	MSU
ARTAN	DPDFQ	MSUS
ATOQX	DPDFQS	MTQ
CROSS	DPDTRM	MXM
CROSSDP	DPEVAL	MXV
DABVAL	DPOLY	ORTHOG
DCROSS	DPOSSO	PEVAL
DDOT	DPRK2	POLY
DETRM	DPRK2S	POSSUM
DEULER	DUNIT	QCONJG
DEULRD	DVXM	QNORM
DEULRE	DXPOSE	QTM
DIFEQ	DXTRCT	QTOAX
DIFEQP	DXYZMT	QUTMLT
DIFEQS	DYZXMT	QXFORM
DIFQPS	EULER	RAP
DMXM	EULERD	SGN
DMXV	EULERE	SIG
DNVRSE	EXTRCT	SIGNM
DORTHO	INVERSE	

ABVAL

Function ABVAL (VECTOR)

ABVAL takes the absolute value of a single-precision three-element vector and returns a single-precision scalar.

ACCOSH (X)

Function ACCOSH (X)

Function ACCOSH (X) returns the single-precision hyperbolic arc cosine of x in radians.

ARSINN

Function ARSINN (THETA)

Single-precision real function ARSINN computes the angle whose sine is equal to THETA. ARSINN calls the system ARCSIN (ASIN) function but does some testing to ensure that ASIN will not abort the run. Angle output is in radians in the interval from $-\frac{\pi}{2}$ to $+\frac{\pi}{2}$.

ARTAN

Function ARTAN (TOP, BOTTOM)

Single-precision real function ARTAN computes the angle whose sine and cosine are equal to TOP and BOTTOM, respectively. The system routine ATAN2 is called if the testing of the input arguments are such that they will not abort the run. Angle output is in radians on the interval from $-\pi$ to $+\pi$.

ATOQX

Subroutine ATOQX (A, Q)

ATOQX extracts a quaternion Q from an Euler sequence of rotations A2, A3, A1 about the Y, Z, X axes, respectively.

CROSS

Subroutine CROSS (VECTR1, VECTR2, VECTR3)

Single-precision subroutine CROSS computes the vector cross product of vector VECTR1 with vector VECTR2, returning the resultant vector in VECTR3.

CROSSDP

Subroutine CROSSDP (VECTR1, VECTR2, VECTR3)

CROSSDP computes VECTR3, the cross product of VECTR1 times VECTR2 (in that order) where all vectors are double precision.

DABVAL

Function DABVAL (DVECT)

Double-precision function DABVAL returns the magnitude of the vector DVECT.

DCROSS

Subroutine DCROSS (DVECT1, DVECT2, DVECT3)

Double-precision subroutine DCROSS computes the vector cross product of the vectors DVECT1 with DVECT2, returning the resultant vector in DVECT3. All quantities are double precision.

DDOT

Function DDOT (X, Y)

Double-precision function DDOT returns the vector dot product of the double precision vectors X and Y.

DETRM

Subroutine DETRM (DETRMT, VALUE)

DETRM evaluates the determinant DETRMT, returning the result in VALUE. DETRMT is a 3x3 matrix stored sequentially by columns.

DEULER

Subroutine DEULER (RPYH, RPYMAT)

Double-precision version of EULER (see EULER). Computes a transformation matrix when given roll, pitch, yaw in that order.

DEULRD

Subroutine DEULRD (EULER, W, DW, DEUL, DDEUL)

Double-precision version of EULERD. Calculates Euler rates and accelerations.

DEULRE

Subroutine EULRE (DT, DEUL, DDEUL, DE)

Double-precision version of EULERE. Calculates double precision estimated delta Euler angle set from Euler rates and accelerations.

DIFEQ

Subroutine DIFEQ (NEQNS, X, DX, Y, IPHASE)

DIFEQ solves a set of first-order differential equations using a four-pass, Runge-Kutta numerical integration technique. NEQNS specifies the number of first-order equations to be integrated. X is the independent variable, and DX is the integration step size. Y is an array, dimensioned 4NEQNS + 1, which contains the dependent variables in the first NEQNS locations, the corresponding derivatives in the second NEQNS locations, and a working area for DIFEQ in the remaining locations. IPHASE is a flag indicating which of the four passes has been executed and is set by DIFEQ. IPHASE is reset to 0 after the pass which completes the integration step. Before the first pass of an integration, the independent variable, dependent variables, derivatives, and the flag must be set. After each space, the derivatives must be reevaluated for the next step.

DIFEQP

Subroutine DIFEQP (NEQNS, X, DX, Y, IPHASE)

DIFEQP is the same as DIFEQ in that it solves a set of first-order differential equations using a four-pass, Runge-Kutta numerical integration technique. All calling arguments have the same definition except for Y, which is an array dimensioned 5NEQNS + 1. DIFEQP differs in technique from DIFEQ in that during the last (fourth) pass of an update, DIFEQP calculates the roundoff due to computer word length and saves this value, which is added back in during the next update. Use and implementation of DIFEQP is the same as DIFEQ, but a more accurate result is obtained..

DIFEQS

Subroutine DIFEQS (NEQNS, X, DX, Y, IPHASE)

DIFEQS is an entry point in DIFEQ and is like DIFEQ, except it is used for second-order equations. NEQNS is equal to twice the number of second-order equations for DIFEQS. Y is an array, dimensioned 4NEQNS + 1, containing the dependent variables in the first NEQNS/2 locations, the first-order derivatives in the second NEQNS/2 locations and in the third NEQNS/2 locations, the second-order derivatives in the fourth NEQNS/2 locations, and a DIFEQS working area in the remainder of the array. The integration process for DIFEQS is the same as for DIFEQ, where the first NEQNS locations for DIFEQS are considered to be dependent variables and the second NEQNS locations are the derivatives.

DIFQPS

Subroutine DIFQPS (NEQNS, X, DX, Y, IPHASE)

DIFQPS is an entry point in DIFEQP and is like DIFEQP, except it is used for integrating second-order equations. Definition of the calling arguments is the same as DIFEQP except for NEQNS, which is equal to twice the number of second-order equations. The Y array assignments are the same as outlined in DIFEQS.

DMXM

Subroutine DMXM (DMAT1, DMAT2, DMAT3)

Double-precision subroutine DMXM computes the product of the matrices DMAT1 with DMAT2 returning the resultant matrix in DMAT3.

DMXV

Subroutine DMXV (DMAT1, DVECT1, DVECT2)

Double-precision subroutine DMXV computes the product of the matrix DMAT1 with the vector DVECT1, returning the resultant vector in DVECT2.

DNVRSE

Subroutine DNVRSE (DMAT1, DMAT2)

Double-precision subroutine DNVRSE computes the inverse of the matrix DMAT1, returning the resultant matrix in DMAT2.

DORTHO

Subroutine DORTHO (AM)

Double-precision version of ORTHOG. Orthogonalizes a matrix, i.e., creates a set of three orthogonal unit vectors from it.

DOT

Function DOT (VECTR1, VECTR2)

DOT returns the vector dot product of VECTR1 and VECTR2.

DOTDP

Function DOTDP (VECTR1, VECTR2)

DOTDP returns the double-precision dot product of two double-precision vectors.

DPDFQ

Subroutine DPDFQ (NEQNS, X, DX, Y, IPHASE)

DPDFQ is the double-precision version of DIFEQ. All calling arguments are single precision except Y, which is a 4NEQNS + 1 dimensioned double-precision array. Use and implementation of DPDFQ is the same as DIFEQ.

DPDFQS

Subroutine DPDFQS (NEQNS, X, DX, Y, IPHASE)

DPDFQS is an entry point in DPDFQ and is like DPDFQ, except it is used for second-order equations. Definition of the calling arguments is the same as DPDFQ except for NEQNS, which is equal to twice the number of second-order equations. The double-precision Y array assignments are the same as outlined in DIFEQS.

DPDTRM

Subroutine DPDTRM (DETRMT, VALUE)

Double-precision subroutine DPDTRM evaluates the double-precision determinant DETRMT, returning the result in double-precision VALUE. DETRMT is a three by three determinant stored sequentially by columns.

DPEVAL

Subroutine DPEVAL (A, N, X, PY)

Double-precision version of PEVAL. DPEVAL evaluates a polynomial of N terms with coefficients A at X and stores the value in PX.

DPOLY

Subroutine DPOLY (X, FX, N, C)

Double-precision version of POLY. DPOLY finds a polynomial of order N-1 to relate N values of independent variable X to N values of dependent variable FX. The coefficients are stored in array C. The number of points, N, must be 25 or smaller.

DPOSSU

Subroutine DPOSSU (TIME, RSMKS, RMMKS)

Double-precision version of POSSUM. Computes position and apparent diameter of sun and moon using data from reference 5.

DPRK2

Subroutine DPRK2 (NEQNS, X, DX, Y, IPHASE)

DPRK2 solves a set of first-order differential equations in double precision, using the second-order, two-pass Runge-Kutta numerical integration technique. NEQNS and IPHASE are integer variables. X, DX, and Y are double-precision variables.

DPRK2 differs from DIFEQ in that it is a second-order solution instead of fourth-order, and the computations are in double precision instead of single precision. The definition of the calling arguments is the same as for DIFEQ, except for precision. DPRK2 specifically uses the "Heun form" second-order Runge-Kutta solution. During the first pass through DPRK2, time is advanced to $X + \frac{2}{3}DX$, and IPHASE is set to 1. During the second pass, time is advanced over the final one-third of the integration step, and IPHASE is set to 0. Prior to each pass, the calling routine is responsible for reevaluating the derivatives.

DPRK2S

Subroutine DPRK2S (NEQNS, X, DX, Y, IPHASE)

DPRK2S is like DPRK2, except it is used for second-order equations. NEQNS and IPHASE are integer variables. X, DX, and Y are double-precision variables. Except for precision, the definition of the calling arguments is the same as for DIFEQS. DPRK2 and DPRK2S have the same relationship as DIFEQ and DIFEQS.

DUNIT

Subroutine DUNIT (DVECT1, DVECT2)

Double-precision subroutine DUNIT unitizes DVECT1, returning the resultant vector in DVECT2.

DVXM

Subroutine DVXM (DVECT1, DMAT1, DVECT2)

Double-precision subroutine DVXM computes the product of DVECT1 with matrix DMAT1, returning the resultant vector in DVECT2.

DXPOSE

Subroutine DXPOSE (DMAT1, DMAT2)

Double-precision subroutine DXPOSE transposes matrix DMAT1, returning the resultant matrix in DMAT2.

DXTRCT

Double-precision version of EXTRCT. Extracts an RPY Euler sequence from an attitude transformation matrix.

DXYZMT

Subroutine DXYZMT (EULERS, XMATRX)

Given the three double-precision Euler angles EULERS (in radians), double-precision subroutine DXYZMT defines the double-precision transformation matrix XMATRX, assuming an X (roll), Y (pitch), Z (yaw) order of rotation. EULERS is an array of roll, pitch, yaw angle sequence.

DYZXMT

Subroutine DYZXMT (EULERS, XMATRX)

Given the three double-precision Euler angles EULERS (in radians), double-precision subroutine DYZXMT defines the double-precision transformation matrix XMATRX, assuming a Y (pitch), Z (yaw), X (roll) order of rotation. EULERS is an array of pitch, yaw, and roll angle sequence.

EULER

ROUTINE DESCRIPTION

The EULER routine calculates a roll, pitch, yaw rotational sequence transformation matrix.

MATH MODEL

The transformation matrix erected by EULER assumes a roll, pitch, yaw rotation sequence. The three angles are received via calling arguments and then substituted into the following equation.

```
RPYMAT(1) = COS(P) COS(Y)
RPYMAT(2) = -COS(P) SIN(Y)
RPYMAT(3) = SIN(P)
RPYMAT(4) = SIN(R) SIN(P) COS(Y) + COS(R) SIN(Y)
RPYMAT(5) = -SIN(R) SIN(P) SIN(Y) + COS(R) COS(Y)
RPYMAT(6) = -SIN(R) COS(P)
RPYMAT(7) = -COS(R) SIN(P) COS(Y) + SIN(R) SIN(Y)
RPYMAT(8) = COS(R) SIN(P) SIN(Y) + SIN(R) COS(Y)
RPYMAT(9) = COS(R) COS(P)
```

where R, P, Y are the roll, pitch, yaw angles.

INPUT/OUTPUT

All input/output of EULER is done via calling arguments.

The call to EULER IS

```
CALL EULER (RPYA, RPYMAT)
```

where

RPYA is the input array consisting of the roll, pitch, and yaw angles in that order.

RPYMAT is the output array consisting of the nine elements of the transformation matrix.

ROUTINE VERIFICATION

This routine was verified by logic review and by comparing routine output to hand calculated values.

EULERD

ROUTINE DESCRIPTION

The EULERD routine calculates the Euler angle rates and accelerations.

MATH MODEL

Given the vehicle body rates and accelerations and the attitude, the Euler angle rates and accelerations can be calculated.

$$\dot{\phi} = \frac{p\cos\psi - q\sin\psi}{\cos\theta}$$

$$\dot{\theta} = p\sin\psi + q\cos\psi$$

$$\dot{\psi} = r - \dot{\phi}\sin\theta$$

where

$$\bar{\omega} = p\hat{i} + q\hat{j} + r\hat{k} = \dot{\bar{\phi}} + \dot{\bar{\theta}} + \dot{\bar{\psi}}$$

and

$$\ddot{\phi} = \frac{(\dot{p} - q\dot{\psi})\cos\psi - (\dot{q} + p\dot{\psi})\sin\psi + \dot{\phi}\dot{\theta}\sin\theta}{\cos\theta}$$

$$\ddot{\theta} = (p\dot{\psi} + \dot{q})\cos\psi + (\dot{p} - q\dot{\psi})\sin\psi$$

$$\ddot{\psi} = \dot{r} - \dot{\phi}\dot{\theta}\cos\theta - \ddot{\phi}\sin\theta$$

The above equations become indeterminate when the pitch angle, θ, is $\pm 90^{\circ}$. To obtain the Euler rates and accelerations at this point, a different set of equations is used.

$$\dot{\phi} = \frac{1}{2\dot{\theta}}\left(-\dot{p}\cos\psi + \dot{q}\sin\psi\right) + \frac{r}{2}$$

$$\dot{\theta} = p\sin\psi + q\cos\psi$$

$$\dot{\psi} = \frac{1}{2\dot{\theta}}\left(\dot{p}\cos\psi - \dot{q}\sin\psi\right) + \frac{r}{2}$$

and

$$\ddot{\phi} = -\frac{1}{3}\left[\dot{\psi}(q\dot{\psi} - 2\dot{p})\frac{\sin\psi}{\dot{\theta}} - \dot{\psi}(p\dot{\psi} + 2\dot{q})\frac{\cos\psi}{\dot{\theta}} + \frac{\ddot{\theta}\dot{\theta}}{\dot{\theta}}\right] + \frac{\dot{r}}{3}$$

$$\ddot{\theta} = (p\dot{\psi} + \dot{q})\cos\psi + (\dot{p} - q\dot{\psi})\sin\psi$$

$$\ddot{\psi} = \dot{r} - \ddot{\phi}$$

Actually, the switch to the above equations for determining the Euler rates and acceleration is made as $\theta \to \pm 90^{\circ}$. Even in the latter set of equations $\dot{\theta}$ can become zero in which case the equations below are used.

$$\dot{\theta} = \pm (p^2 + q^2)^{\frac{1}{2}}$$

$$\dot{\phi} = \frac{1}{2}\left[r - (\dot{p}q - \dot{q}p)/(p^2 + q^2)\right]$$

$$\dot{\psi} = \frac{1}{2}\left[r + (\dot{p}q - \dot{q}p)/(p^2 + q^2)\right]$$

and

$$\ddot{\theta} = (p\dot{\psi} + \dot{q})\cos\psi + (\dot{p} - q\dot{\psi})\sin\psi$$

$$\ddot{\phi} = \frac{1}{3}\left\{\dot{r} - \left[\dot{\psi}(q\dot{\psi} - 2\dot{p})p - \dot{\psi}(p\dot{\psi} + 2\dot{q})q\right]/(p^2 + q^2) - \frac{\ddot{\theta}\dot{\phi}}{\dot{\theta}}\right\}$$

$$\ddot{\psi} = \dot{r} - \ddot{\phi}$$

The rotational sequence assumed in deriving these equations is roll, pitch, yaw.

The definitions of the terms in the above equations are:

p,q,r - vehicle body angular rates

$\dot{p},\dot{q},\dot{r}$ - vehicle body angular accelerations

ϕ,θ,ψ - Euler angles roll, pitch, yaw

$\dot{\phi},\dot{\theta},\dot{\psi}$ - Euler angle rates

$\ddot{\phi},\ddot{\theta},\ddot{\psi}$ - Euler angle accelerations

INPUT/OUTPUT

All input and output of EULERD is via calling arguments. The input arguments are:

ϕ,θ,ψ - Euler angles roll, pitch, yaw [PS: EULER]

p,q,r - Vehicle body angular rates [PS: W]

$\dot{p},\dot{q},\dot{r}$ - Vehicle body angular accelerations [PS: DW]

The output arguments are:

$\dot{\phi},\dot{\theta},\dot{\psi}$ - Euler angle rates [PS: DEUL]

$\ddot{\phi},\ddot{\theta},\ddot{\psi}$ - Euler angle accelerations [PS: DDEUL]

where [PS:] indicates the program symbol.

ROUTINE VERIFICATION

This routine was verified by logic review and by comparing routine output to hand calculated values.

EULERE

ROUTINE DESCRIPTION

The EULERE routine calculates an estimated delta Euler angle set from Euler rates and accelerations.

MATH MODEL

The previously calculated Euler angle rates and accelerations are used to estimate the Euler angles change over some delta time.

$$\Delta c_i = \dot{c}_i \Delta T + \frac{1}{2}\ddot{c}_i \Delta T^2$$

where

Δc_i = is the Euler angle change over the ΔT. [PS: DE]

$\dot{c}_i$ = is the Euler angle rates [PS: DEUL]

$\ddot{c}_i$ = is the Euler angle accelerations [PS: DDEUL]

ΔT = is the delta time period [PS: DT]

INPUT/OUTPUT

All input/output of EULERE is done via calling arguments.

The call to EULERE is

```
CALL EULERE (DT,DEUL,DDEUL,DE)
```

where the arguments are defined above in [PS:].

ROUTINE VERIFICATION

This routine was verified by logic review and by comparing routine output to hand calculated values.

EXTRCT

ROUTINE VERIFICATION

The EXTRCT routine extracts from an attitude matrix a set of three Euler angles assuming a roll, pitch, yaw rotational sequence.

MATH MODEL

The EXTRCT routine assumes the attitude matrix was formed using a roll, pitch, yaw rotation sequence and to have the following form:

$$\begin{bmatrix} \cos(P)\cos(Y) & \sin(R)\sin(P)\cos(Y) + \cos(R)\sin(Y) & -\cos(R)\sin(P)\cos(Y) + \sin(R)\sin(Y) \\ -\cos(P)\sin(Y) & -\sin(R)\sin(P)\sin(Y) + \cos(R)\cos(Y) & \cos(R)\sin(P)\sin(Y) + \sin(R)\cos(Y) \\ \sin(P) & -\sin(R)\cos(P) & \cos(R)\cos(P) \end{bmatrix}$$

where R,P,Y are the roll, pitch, yaw Euler angles.

To find the pitch angle, the arcsine is taken of matrix element 3,1. The yaw angle can be found by dividing the matrix elements 1,1 or 2,1 by the cosine of the pitch angle and then taking the arccosine or arcsine respectively. The roll angle is found by the same procedure using matrix elements 3,2 and 3,3.

As the pitch angle approaches $\pm 90°$, the matrix approaches a singularity. When this situation arises, the delta Euler angles calculated by EULERE are added to the old values to obtain an estimated value.

INPUT/OUTPUT

All input/output of EXTRCT is done via calling arguments. The call to EXTRCT is

```
CALL EXTRCT (AM,DE,A)
```

where inputs are:

AM - is the attitude matrix
DE - is the delta Euler angles

output is:

A - is the attitude array of roll(R), pitch(P), and yaw(Y) angles.

ROUTINE VERIFICATION

This routine was verified by logic review and by comparing routine output to hand calculated values.

INVRSE

Subroutine INVRSE (XMATRX, YMATRX)

INVRSE computes the inverse of matrix XMATRX, returning the result in matrix YMATRX.

MATYPR

ROUTINE DESCRIPTION

The MATYPR routine extracts from an attitude matrix a set of three Euler angles assuming a yaw, pitch, roll rotational sequence.

MATH MODEL

The MATYPR routine assumes the attitude matrix was formed using a yaw, pitch, roll rotation sequence and having the following form:

$$\begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix}$$

where ψ, θ, ϕ are the yaw, pitch, roll Euler angles.

The pitch angle is calculated as follows:

$$P = \text{ASIN}(-\sin\theta)$$

The yaw angle is calculated as

$$Y_1 = \frac{\cos\theta\cos\psi}{\cos P}$$

or

$$Y_2 = \frac{\cos\theta\sin\psi}{\cos P}$$

and finally

$$Y = \text{ACOS } Y_1$$

or

$$Y = \text{ASIN } Y_2$$

Likewise, the roll angle is calculated as

$$R_1 = \frac{\cos\theta\sin\phi}{\cos P}$$

or

$$R_2 = \frac{\cos\theta\cos\phi}{\cos P}$$

where

$$R = \text{ASIN } R_1$$

$$R = \text{ACOS } R_2$$

where Y, P, R are the yaw, pitch, roll Euler angles.

As the pitch angle approaches $\pm 90^0$, the matrix approaches a singularity. When this situation arises, the delta Euler angles calculated by EULERE are added to the old values to obtain an estimated value (ref. 4).

INPUT/OUTPUT

All input/output of MATYPR is done via calling arguments.

The call to MATYPR is

CALL MATYPR (AM, DE, A)

where inputs are:

AM is the attitude matrix.

DE is the delta Euler angles.

Output is:

A is the attitude array of yaw (Y), pitch (P), and roll (R) angles.

ROUTINE VERIFICATION

This routine was verified by logic review and by comparing routine output to previously computed values.

SOURCE

MATYPR was derived primarily from reference 4.

MSU

Subroutine MSU (ANGLS1, ANGLS2, ANGLS3)

MSU performs the angular modular subtraction of the angle sets ANGLS1, ANGLS2, returning the results in ANGLS3. ANGLS1, ANGLS2, and ANGLS3 are dimensioned three and are in radians. ANGLS3 are angles in the range $\pm\pi$ with a counterclockwise rotation positive.

MSUS

Subroutine MSUS (ANGLS1, ANGLS2, ANGLS3)

MSUS performs the angular modular subtraction of the single angles ANGLS1 and ANGLS2. The resulting angle is returned in ANGLS3. ANGLS1, ANGLS2, and ANGLS3 are single angles and are in radians. ANGLS3 is an angle in the range $\pm\pi$ with a counterclockwise rotation positive.

MTQ

Subroutine MTQ (M, Q)

MTQ extracts a quaternion from a direction cosine matrix. The extraction process avoids singularities. A positive scalar part is arbitrarily chosen for the resulting quaternion. It is assumed that the input matrix is unitary.

MXM

Subroutine MXM (XMATRX, YMATRX, ZMATRX)

MXM computes the product of matrix XMATRX with YMATRX, returning the resultant matrix in ZMATRX.

MXV

Subroutine MXV (XMATRIX, VECTR1, VECTR2)

MXV computes the product of matrix XMATRX with vector VECTR1, returning the resultant vector in VECTR2.

ORTHOG

ROUTINE DESCRIPTION

The ORTHOG routine orthogonalizes an attitude matrix.

MATH MODEL

The attitude matrix [AM] is assumed to be defined by

$$[AM] = \begin{bmatrix} \overline{AM}_1 \\ \overline{AM}_4 \\ \overline{AM}_7 \end{bmatrix}$$

To orthogonalize [AM] it is first transposed.

$$[AMT] = [AM]^T = [\overline{AM}_1, \overline{AM}_4, \overline{AM}_7]$$

Then,

$$\overline{AMT}_7 = \overline{AMT}_1 \times \overline{AMT}_4$$

The vector components $\overline{AMT}_7$ and $\overline{AMT}_1$ are unitized, and

$$\overline{AMT}_4 = \overline{AMT}_7 \times \overline{AMT}_1$$

The matrix [AMT] is now transposed back to [AM].

INPUT/OUTPUT

The matrix [AM] is input, orthogonalized and then output through one calling argument to ORTHOG.

ROUTINE VERIFICATION

This routine was verified by comparing routine output to hand calculated values.

PEVAL

ROUTINE DESCRIPTION

PEVAL evaluates a given polynomial at a given point. Used in conjunction with the curve-fitting utility routine POLY, PEVAL provides a means of extracting current values from tabulated input.

MATH MODEL

The polynomial

$$P(x) = a_o + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

is evaluated by Horner's method, defined recursively as:

$$P_o(x) = a_{n-1}$$

$$P_i(x) = P_{i-1}(x) \cdot x + a_{n-1-i}$$

with $P(x) = P_{n-1}(x)$

INPUT/OUTPUT

Subroutine PEVAL is referenced by

```
CALL PEVAL(A,N,X,PX)
```

with arguments defined as:

- A - array of polynomial coefficients: $A(I) = a_{I-1}$
- N - dimension of A (N = degree (P) + 1)
- X - independent variable for which polynomial is to be evaluated
- PX - resultant value of dependent variable

PEVAL uses no COMMON input, contains no internal data, and makes no external references.

ROUTINE VERIFICATION

Output from PEVAL was compared with hand-calculated values.

POLY

ROUTINE DESCRIPTION

POLY fits a polynominal curve to a set of tabulated data points. Used in conjunction with the polynominal-evaluation utility routine PEVAL, POLY provides a means of extracting current values from tabulated input.

MATH MODEL

Given n data points $(x_i, f(x_i))$, POLY approximates the function f with the unique polynomial of degree n-1 or less defined by

$$P(x) = \sum_{i=1}^{n} f(x_i)\ L_i(x) \qquad (1)$$

where L_i is the ith Lagrangian polynomial

$$L_i(x) = \frac{(x-x_1)\ldots(x-x_{i-1})\ (x-x_{i+1})\ldots(x-x_n)}{(x_i-x_1)\ldots(x_i-x_{i-1})\ (x_i-x_{i+1})\ldots(x_i-x_n)}$$

This polynomial will duplicate the function f at the specified data points (except for round-off error), but may vary from f between input points (as does linear interpolation). Since intermediate values in the computation of the Lagrangian polynomials may become many orders of magnitude larger than the input values, double precision variables are used internally to avoid floating point overflow and underflow. However, double precision overflow is still a consideration. If POLY is called with too many data points, overflow and/or underflow may cause divergence from the input function f, in which case POLY should be called with fewer points specified.

POLY returns the polynomial P(x) as defined by equation 1 in the form

$$P(x) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1} \qquad (2)$$

INPUT/OUTPUT

Subroutine POLY is referenced by

```
CALL POLY(X,FX,N,C)
```

with arguments defined as:

- X - an array of N distinct values of the independent variable; their order is irrelevant, but no two may be equal.
- FX - an array of the N values of the dependent variable, corresponding by index to X - i.e., FX(I) = f(X(I))
- N - the number of data points specified.
- C - the output array of coefficients of the returned polynomial: C(I) = a_{I-1} in equation 2.

POLY uses no COMMON input and makes no external references.

Internal arrays are dimensioned 25; this value must be changed if POLY is to be called with more than 25 data points. (This practice is not recommended due to the overflow considerations mentioned above).

ROUTINE VERIFICATION

POLY was checked out both independently and in conjunction with utility routine PEVAL, with output coefficients and values compared with hand-calculated results.

POSSUM/80

Versions: /80SP single precision
/80DP double precision

POSSUM is a routine which computes the earth-centered-inertial (ECI) Cartesian positions of the sun and moon. It is meant to be called either from the gravity force model GRAV2 (with lunar and solar perturbation flags on) or from the Relative Attitude Processor (RAP), a flight software routine. POSSUM is designed to use orbital data for the sun and moon as published in the American Ephemeris and Nautical Almanac (ref. 5) in years up to and including 1980. This data is in the form of orbital elements; semimajor axis, a, eccentricity, e, inclination, i, mean motion, M, longitude of ascending mode, Ω, longitude of perigee, ω, and longitude of sun or moon at epoch, ℓ.

Sample orbital element data is displayed in figure 3-6. The POSSUM requires that the epoch for the elements be input in the form of year since 1900, month (two digits), day (two digits). The date for the start of the simulation is needed in the same form. This routine converts these dates into day counts since 1900 for both the epoch and the date of simulation. The difference between the epoch and the date of simulation is computed and a warning issued if it is inconsistent with the availability of new orbital data (elements are updated roughly monthly). The fraction of a day at the start of the simulation is then added to the number of days elapsed since epoch. Integer numbers of lunar and/or solar periods are removed from this elapsed time.

On the first pass through POSSUM, the epoch is used as a reference time; on successive passes, the time at the last call is used for a reference. In either case, the true anomaly at epoch is computed using $\nu_o = \ell - \omega$. The eccentric anomaly at the corresponding time is computed via (ref. 6)

$$\cos E_o = \frac{e + \cos \nu_o}{1 + e \cos \nu_o} \tag{1}$$

Kepler's equation is iterated using

$$\Delta E = \Delta tM - (E_t + e \sin E_t) + (E_o + e \sin E_o) \tag{2}$$

MOON, 1980

MEAN ELEMENTS OF MOON'S EQUATOR AND ORBIT

Date	Mean Equator of Moon referred to true equator of Earth			Mean Orbital Elements referred to ecliptic of date			
				Mean Longitude			Elongation
IDATE(1)	I	Δ	Ω'	Perigee OMEGA(2)	Node XOMEGA(2)	Moon XLONGO(1)	from Sun
	°	°	°	°	°	°	°
Jan. −8	24·869	333·920	−1·697	348·4919	152·3741	319·5644	48·6160
2	24·803	333·419	1·727	349·6060	151·8445	91·3283	170·5235
12	24·796	332·917	1·757	350·7200	151·3150	223·0923	292·4310
22	24·790	332·415	1·787	351·8340	150·7854	354·8563	54·3385
Feb. 1	24·783	331·912	1·817	352·9481	150·2559	126·6202	176·2460
11	24·777	331·410	−1·847	354·0621	149·7264	258·3842	298·1535
21	24·770	330·908	1·876	355·1761	149·1968	30·1482	60·0610
Mar. 2	24·763	330·405	1·906	356·2902	148·6673	161·9121	181·9685
12	24·756	329·902	1·935	357·4042	148·1378	293·6761	303·8759
22	24·749	329·399	1·965	358·5183	147·6082	65·4401	65·7834
Apr. 1	24·742	328·896	−1·994	359·6323	147·0787	197·2040	187·6909
11	24·734	328·393	2·023	0·7463	146·5491	328·9680	309·5984
21	24·727	327·890	2·051	1·8604	146·0196	100·7320	71·5059
May 1	24·719	327·386	2·080	2·9744	145·4901	232·4959	193·4134
11	24·711	326·883	2·109	4·0884	144·9605	4·2599	315·3209
21	24·703	326·379	−2·137	5·2025	144·4310	136·0239	77·2284
31	24·695	325·875	2·165	6·3165	143·9014	267·7878	199·1359
June 10	24·687	325·372	2·193	7·4305	143·3719	39·5518	321·0434
20	24·679	324·868	2·221	8·5446	142·8424	171·3157	82·9509
30	24·671	324·364	2·249	9·6586	142·3128	303·0797	204·8584
July 10	24·663	323·859	−2·276	10·7727	141·7833	74·8437	326·7658
20	24·655	323·355	2·304	11·8867	141·2538	206·6076	88·6733
30	24·646	322·850	2·331	13·0007	140·7242	338·3716	210·5808
Aug. 9	24·638	322·345	2·358	14·1148	140·1947	110·1356	332·4883
19	24·629	321·840	2·385	15·2288	139·6651	241·8995	94·3958
29	24·620	321·335	−2·411	16·3428	139·1356	13·6635	216·3033
Sept. 8	24·611	320·830	2·438	17·4569	138·6061	145·4275	338·2108
18	24·602	320·324	2·464	18·5709	138·0765	277·1914	100·1183
28	24·593	319·818	2·490	19·6849	137·5470	48·9554	222·0258
Oct. 8	24·584	319·313	2·516	20·7990	137·0174	180·7194	343·9333
18	24·575	318·807	−2·542	21·9130	136·4879	312·4833	105·8408
28	24·565	318·300	2·568	23·0271	135·9584	84·2473	227·7482
Nov. 7	24·556	317·794	2·593	24·1411	135·4288	216·0113	349·6557
17	24·546	317·288	2·618	25·2551	134·8993	347·7752	111·5632
27	24·536	316·781	2·643	26·3692	134·3698	119·5392	233·4707
Dec. 7	24·527	316·274	−2·668	27·4832	133·8402	251·3031	355·3782
17	24·517	315·767	2·693	28·5972	133·3107	23·0671	117·2857
27	24·507	315·260	2·717	29·7113	132·7811	154·8311	239·1932
37	24·497	314·753	−2·741	30·8253	132·2516	286·5950	1·1007
	Daily motion			+0°·111404	−0°·052954	13°·176396	12°·190749

Epoch 1900 January 0·5 E.T.

Eccentricity = 0·05490 0489 = ECC(2)

Inclination = 5°·145 3964 = XINC(2)

Figure 3-6.- Correspondence of tabular data and orbital elements.

INNER PLANETS, 1980

MEAN ELEMENTS FOR MEAN EQUINOX AND ECLIPTIC OF DATE

Epoch 1980 June 10·0 : J.D. 244 4420·5 ; variations for 100 days

Planet	XINC(2) Inclination i	var.	XOMEG(2) Longitude of Ascending Node Ω	var.	XOMEG(2)+180° Longitude of Perihelion ϖ	var.	Mean Distance a	XMMOTN STD Mean Motion n	ECC(1) Eccentricity e
	°		°		°			°	
Mercury	7·004 37	+1	48·009 43	+325	77·151 12	+426	0·387 098	4·092 339	0·205 641
Venus	3·394 44	0	76·503 75	+247	131·295 82	+385	0·723 332	1·602 130	0·006 783
Earth	...	...	...	...	102·663 02	+471	1·000 000	0·985 609	0·016 717
Mars	1·849 80	0	49·466 62	+211	335·098 58	+504	1·523 691	0·524 033	0·093 387

Date	Julian Date	Mean Longitude SUN XLONGO(1)	Mean Anomalies MERCURY	VENUS	EARTH	MARS
		°	°	°	°	°
Jun. −8	244 4230·5	270·9484	121·4144	211·6269	348·3523	146·4381
2	4240·5	280·8048	162·3378	227·6482	358·2083	151·6784
12	4250·5	290·6613	203·2611	243·6695	8·0644	156·9186
22	4260·5	300·5178	244·1845	259·6908	17·9204	162·1588
Feb. 1	4270·5	310·3743	285·1078	275·7121	27·7764	167·3990
11	244 4280·5	320·2307	326·0312	291·7334	37·6324	172·6392
21	4290·5	330·0872	6·9545	307·7547	47·4884	177·8794
Mar. 2	4300·5	339·9437	47·8779	323·7760	57·3444	183·1196
12	4310·5	349·8002	88·8012	339·7973	67·2004	188·3598
22	4320·5	359·6566	129·7245	355·8186	77·0564	193·6000
Apr. 1	244 4330·5	9·5131	170·6479	11·8399	86·9124	198·8402
11	4340·5	19·3696	211·5712	27·8612	96·7684	204·0804
21	4350·5	29·2260	252·4946	43·8825	106·6244	209·3206
May 1	4360·5	39·0825	293·4179	59·9038	116·4804	214·5608
11	4370·5	48·9390	334·3413	75·9251	126·3364	219·8011
21	244 4380·5	58·7955	15·2646	91·9464	136·1924	225·0413
31	4390·5	68·6519	56·1880	107·9677	146·0484	230·2815
June 10	4400·5	78·5084	97·1113	123·9890	155·9044	235·5217
20	4410·5	88·3649	138·0346	140·0103	165·7604	240·7619
30	4420·5	98·2214	178·9580	156·0316	175·6164	246·0021
July 10	244 4430·5	108·0778	219·8813	172·0529	185·4724	251·2423
20	4440·5	117·9343	260·8047	188·0742	195·3284	256·4825
30	4450·5	127·7908	301·7280	204·0956	205·1844	261·7227
Aug. 9	4460·5	137·6473	342·6514	220·1169	215·0404	266·9629
19	4470·5	147·5037	23·5747	236·1382	224·8964	272·2031
29	244 4480·5	157·3602	64·4980	252·1595	234·7524	277·4433
Sept. 8	4490·5	167·2167	105·4214	268·1808	244·6084	282·6835
18	4500·5	177·0731	146·3447	284·2021	254·4644	287·9238
28	4510·5	186·9296	187·2681	300·2234	264·3204	293·1640
Oct. 8	4520·5	196·7861	228·1914	316·2447	274·1764	298·4042
18	244 4530·5	206·6426	269·1148	332·2660	284·0324	303·6444
28	4540·5	216·4990	310·0381	348·2873	293·8884	308·8846
Nov. 7	4550·5	226·3555	350·9615	4·3086	303·7444	314·1248
17	4560·5	236·2120	31·8848	20·3299	313·6004	319·3650
27	4570·5	246·0685	72·8081	36·3512	323·4564	324·6052
Dec. 7	244 4580·5	255·9249	113·7315	52·3725	333·3124	329·8454
17	4590·5	265·7814	154·6548	68·3938	343·1684	335·0856
27	4600·5	275·6379	195·5782	84·4151	353·0244	340·3258
37	244 4610·5	285·4944	236·5015	100·4364	2·8804	345·5660
Daily motion		0°985 647	4°092 334	1°602 130	0°985 600	0°524 021

STD = SEC TO DAYS = 86400

Figure 3-6.- Concluded.

where E_t is a trial value of the eccentric anomaly.

When ΔE falls below some tolerance, E_t is adopted and a true anomaly is computed via

$$\cos \nu = \frac{e - \cos E}{e \cos - 1} \tag{3}$$

At this point the distance to the sun or moon is computed assuming an elliptical orbit. The Cartesian geocentric equatorial position components are obtained by first finding ecliptic Cartesian coordinates (ECC)

$$\begin{aligned} X_{ECC} &= (\cos \ell_e \cos \lambda_e) R \\ Y_{ECC} &= (\sin \ell_e \cos \lambda_e) R \\ Z_{ECC} &= (\sin \lambda_e) R \end{aligned} \tag{4}$$

where

λ_e is the ecliptic latitude

ℓ_e is the ecliptic longitude

R is the distance from the origin

Figure 3-7 displays the geometry. From the law of sines

$$\sin \lambda_e = \sin i \sin (\nu + \omega) \tag{5}$$

while from the law of cosines

$$\ell_e = \cos^{-1} \left[\cos (\nu + \omega)/\cos \lambda\right] + \Omega \tag{6}$$

(see fig. 3-7) and

$$R = \frac{a (1 - e^2)}{1 + e \cos \nu} \tag{7}$$

The ECC coordinates are converted to ECI by rotating about the x-axis by ε, the obliquity of the ecliptic. The rotation matrix is

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varepsilon & -\sin \varepsilon \\ 0 & \sin \varepsilon & \cos \varepsilon \end{bmatrix} \tag{8}$$

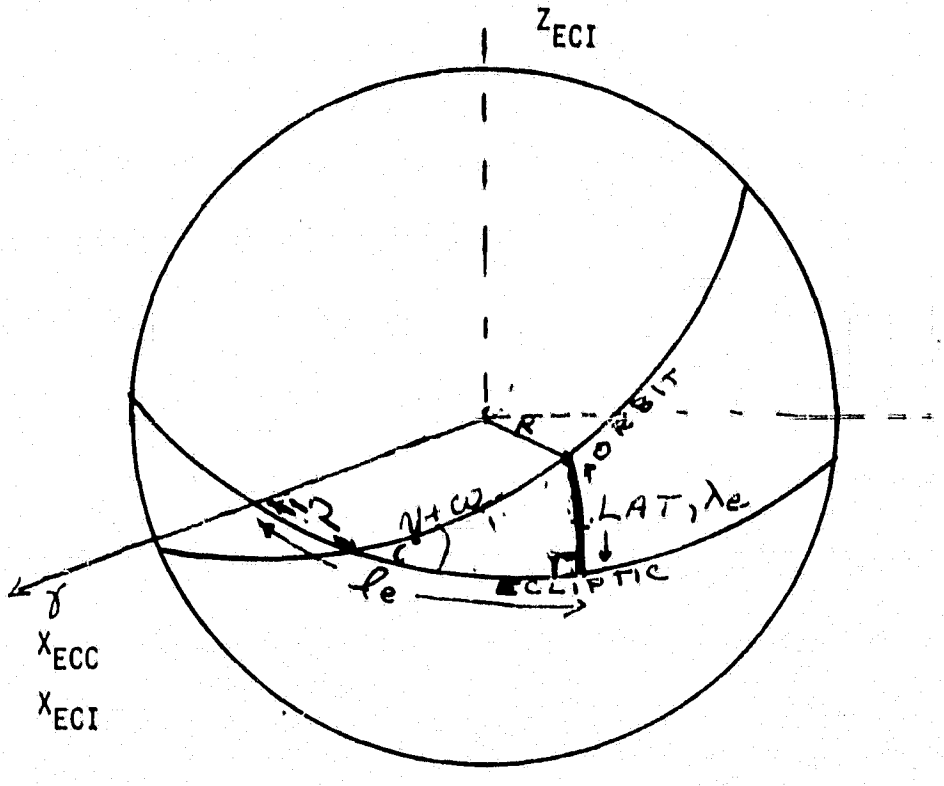


Figure 3-7.- Geometry of ecliptic Cartesian coordinates at the lunar or solar location.

and

$$\begin{bmatrix} X_{ECI} \\ Y_{ECI} \\ Z_{ECI} \end{bmatrix} = B \begin{bmatrix} X_{ECC} \\ Y_{ECC} \\ Z_{ECC} \end{bmatrix} \tag{9}$$

The solar coordinates obtained from equation (9) are relative to the barycenter of the earth-moon system. They are corrected back to the center of the earth by

$$\begin{aligned} X_s &= X_s - X_M/\mu \\ Y_s &= Y_s - Y_M/\mu \\ Z_s &= Z_s - Z_M/\mu \end{aligned} \tag{10}$$

where

μ is the earth-to-moon mass ratio

s denotes solar

M denotes lunar

Unit vectors from the center of the earth to the sun and moon are then constructed. The apparent diameters of the sun and moon are also computed.

This routine has been checked by comparing the resultant values with tabulated apparent semidiameters and directions (right ascension and declination).

INPUT

Program symbol	Common block name	Common block location	Dimen.	Description (units)
TIME	(Calling argument)			Time start of simulation (sec)
TLAST	GRAVC	11		Time at which POSSUM was last called starting from the beginning of the simulation, computed internally after first pass (sec)
IDATE	GRAVC	55 56	2	Epoch for which the orbital elements of sun and moon are defined, then date of simulation. Both dates are in the format: year (two digits), month (two digits), day (two digits). (YYMMDD)
FDATE	GRAVC	57		Fraction of a day at start of simulation (days)
A	GRAVC	58 59	2	Semimajor axes of sun and then moon geocentric orbits (m)
ECC	GRAVC	60 61	2	Eccentricity of solar then lunar, geocentric orbit
XINC	GRAVC	62 63	2	Inclination of solar then lunar, orbit relative to the ecliptic (deg)
XMMOT	GRAVC	64 65	2	Mean motion of sun, then moon, in orbit (deg/sec)
DIAM	GRAVC	66 67	2	Apparent diameter of sun, then moon, at a distance equal to orbital semi-major axis (deg)
XOMEG	GRAVC	68 69	2	Longitude of the ascending mode of orbit of sun, then moon, measured in the plane of the ecliptic (deg)
WOMEG	GRAVC	70 71	2	Longitudes of perigee for sun, then moon, measured from vernal equinox to node in the ecliptic plane, from node to perigee in orbital plane (deg)
XLONGO	GRAVC	72 73	2	Longitude at epoch of sun, then moon (longitude of the sun differs by 180° from longitude of the earth)(deg)

Program symbol	Common block name	Common block location	Dimen.	Description (units)
EL	GRAVC	74 75	2	Lower limit of eccentric anomaly for sun, then moon (rad)
EU	GRAVC	76 77	2	Upper limit of eccentric anomaly for sun, then moon (rad)
UV	GRAV	78-86	3,3	Unit vectors from center of earth to sun (UV(1,1) - UV(3,1)) and to moon (UV(1,2) - UV(3,2))
IFIRST	GRAVC	87		First pass flag
IMNTH	GRAVC	88-99	12	Number of the day of the year on which each month starts
OBLQ	GRAVC	100		Obliquity of the ecliptic (deg)
EMMR	GRAVC	101		Earth-to-moon mass ratio

OUTPUT

Program symbol	Common block name	Common block location	Dimen.	Description (units)
RSMKS	(Calling argument)		3	Position of sun in ECI coordinates (m)
RMMKS	(Calling argument)		3	Position of moon in ECI coordinates (m)
GPOS	GRAVC	12 — 17	6	GPOS(1) = RSMKS(1) GPOS(4) = RMMKS(1) These locations are used to store positions previously computed (m)
UV	GRAVC	78 — 86	3,3	Unit vectors describing the position of the sun, then the moon
APDIAM	GRAVC	102 — 104	3	Apparent diameter of sun then moon, then earth (deg)
R	GRAVC	105	2	Distance of sun, then moon from earth (m)

POSSUM/81

Version: /81SP single precision
/81DP double precision

POSSUM is a routine which computes the ECI Cartesian coordinates of the sun and the moon. It also computes unit vectors to the sun and moon and apparent diameters for each in degrees. This information is meant for use by the gravity force model GRAV2 and by the flight control utility routine RAP. The POSSUM routines use locations in the common block GRAV to store output.

POSSUM stores an epoch for solar positioning (year, month, day) and a starting time for the simulation (year, month, day and fraction of a day in Universal time). On the first pass it computes the number of days elapsed between the epoch for solar constants and the time of the start of the simulation. At times subsequent to the start of the simulation, the time between the epoch and the Universal time at that point in the simulation is computed (in days). The low precision formulae for solar coordinates given in section D of The Astronomical Almanac (ref. 7) are used. These formulae are expected to be accurate to 0.01^{o} in right ascension and declination.

Defining

G = mean anomaly of sun

d = time elapsed since epoch

$SC_{(i)}$ = constant from Astronomical Almanac (ref. 7)

ε = obliquity of the ecliptic

λ = ecliptic longitude of the sun

The computation proceeds as

$$L = SC_{(1)} + SC_{(2)}d$$

$$G = SC_{(3)} + SC_{(4)}d$$

$$\lambda = L + SC_{(5)} \sin(G) + SC_{6} \sin(2G)$$

$$R = 1 - SC_{(7)} \cos(G)$$

$$x = R \cdot A \cos(\lambda)$$

$$y = R \cdot A - \cos(\varepsilon) \sin\lambda$$

$$z = R \cdot A \sin(\varepsilon) \sin\lambda$$

where A is the semimajor axis of the earth's orbit in meters.

The apparent diameter, AD, is found from the diameter at distance A and

$$AD = (A/R) \cdot DIAM(1)$$

where DIAM is the diameter of the sun when A = R.

It should be noted that this formulation does not include any variation of the angular velocity of the sun in its computation.

The position and diameter of the moon are computed using the daily polynominal coefficients for the moon. These coefficients form fifth-degree expressions for the right ascension, RA, declination, δ, and horizontal parallax, π, of the moon in terms of the difference in ephemeris time (ET) between the epoch (0 hr ET of the date in question) and the simulation time. This time difference is found by adding the fraction of a day at which the simulation starts to the time since the start of the simulation and also summing UTET, the correction from Universal time to Ephemeris time.

Once values of RA, δ, and π are found, they are converted to ECI Cartesian coordinates relative to the true equinox of date by

$$R = Re/\pi$$
$$x = R \cos \delta \cos (RA)$$
$$y = R \cos \delta \sin (RA)$$
$$z = R \sin (\delta)$$
$$AD = (A(2)/R)\ DIAM(2)$$

where

A(2) is the semimajor axis of the moon's orbit, in meters

DIAM(2) is the apparent diameter of the moon when A = R.

INPUT

Prog. symbol	COMMON block name	COMMON block location	Dimen.	Description (unit)
TIME	(calling argument)			Time since start of simulation (sec)
SL	GRAVC	55	6,3	Constants for computing lunar position and horizontal parallax. The constants for right ascension are in positions SL(J, 1), for declination in SL(J, 2), and for horizontal parallax in SL(J, 3). The position SL(1, 1) holds the a_5 coefficient for right ascension. SL(1, 3) = 0, always (= a_5 for horizontal parallax)
A	GRAVC	76	2	Semimajor axis of solar then lunar orbit (m)
IDATE	GRAVC	87	2	Epoch of solar constants then date of simulation in form of year, month, day for a total of six digits
FDATE	GRAVC	89		Fraction of a day elapsed from 0 hr UT at the start of the simulation.
DIAM	GRAVC	90	2	Apparent diameter of sun then moon when seen from distance A (deg)
SC	GRAVC	93	7	Solar constants available from the Astronomical Almanac. See equations (2 through 8) for use.
OBLQ	GRAVC	100		Obliquity of the ecliptic (deg)
UTET	GRAVC	101		Correction to add to UT to obtain ET (sec)

OUTPUT

Prog. symbol	COMMON block name	COMMON block location	Dimen.	Description (unit)
RSMKS	(calling argument)			Solar position, ECI (m)
RMMKS	(calling argument)			Lunar position, ECI (m)
GPOS	GRAVC	12	6	RSMKS then RMMKS
R	GRAVC	74	2	Distance of sun then moon from center of earth in meters.
UV	GRAVC	78	3,3	Unit vectors from center of earth sun (UV(1,1) to UV(3,1) and to moon (UV(1,2) to UV(3,2))
APDIAM	GRAVC	102	3	Apparent diameter of sun then moon as seen from the center of the earth (deg)

GRAVC COMMON BLOCK LOCATIONS USED

Program Symbol	COMMON block Location	Dimen.	Description (units)
TLAST	11		Store simulation time of last call to POSSUM (sec)
GPOS	12-18	6	Stores ECI position of sun then moon (m)
SL	55	6,3	Constants for computing right ascension, declination, and horizontal parallax of moon
IFIRST	73		First pass flag
R	74	2	Distance of sun then moon from center of earth (m)
A	76	2	Semimajor axes of orbits of sun then moon (m)
UV	78	3,3	Unit vectors from center of earth to sun then to moon
IDATE	87	2	Epoch of solar constants then date of simulation. Each is in six-digit year, month, day form.
FDATE	89		Fraction of a day elapsed since 0 hr UT at the start of the simulator
DIAM(2)	90		Apparent diameter of sun then moon as seen from distance A (see loc. 76) (deg)
SC	93	7	Constants for computing position of sun (see eqs. (2-8))
OBLQ	100		Obliquity of the ecliptic (deg)
UTET	101		Correction to add to UT to obtain ET (sec)
APDIAM	102	3	Apparent diameter of sun then moon as seen from the center of the earth (deg)
COBL	105		COS (OBLQ)
SOBL	106		SIN (OBLQ)

Note: This routine is designed to be used with the Relative Attitude Processor (RAP). For this reason some of the COMMON block locations are left blank. They will be used by RAP and should not be eliminated.

QCONJG

Subroutine QCONJG (Q, QCONJ)

QCONJG takes the conjugate QCONJ of the quaternion Q.

QNORM

Subroutine QNORM (Q)

QNORM yields a properly normalized quaternion with a positive scalar part. If the incoming scalar is negative, QNORM complements the vector part. QNORM computes the scalar part based on the vector part. Numerical problems in the vicinity of 180° are avoided, and the resulting algorithm corresponds to an exact 180° rotation.

QTM

Subroutine QTM (Q, AMAT)

QTM generates a direction cosine matrix AMAT from its associated quaternion Q.

QTOAX

Subroutine QTOAX (Q, PSI, THETA, PHI, SINTH, COSTH, COSPHI, SINPHI)

QTOAX extracts yaw, pitch, roll sequence Euler angles (PSI, THETA, PHI) from quaternion Q. For THETA = $\pm 90^{\circ}$, PHI is arbitrarily chosen to be 0°. PSI is then extracted so that (PSI, THETA, 0) represents the same attitude as the quaternion Q. The sine and cosine of THETA and PHI are also returned. The angles are returned in radians.

QUTMLT

Subroutine QUTMLT (PP, NP, QQ, NQ, R)

QUTMLT generates the product R of two quaternions, PP and QQ. The order of the input quaternions is important, since in general quaternion multiplication is not commutative. If NP is negative, then the conjugate of PP is used in the product. If NQ is negative, the conjugate of QQ is used in the product.

QXFORM

Subroutine QXFORM (U, QQ, N, V)

QXFORM transforms the vector U into the vector V via the transformation quaternion QQ, thus V=QQ*U*QCONJ. If N is negative, the conjugate of QQ is used.

RELATIVE ATTITUDE PROCESSOR (RAP)

PROGRAM DESCRIPTION

The RAP models a designated optical system and associated field of view (FOV) and determines whether a specified object is within the optical FOV. The routine operates on a pair of systems (vehicles) as defined by the user with each system having one or more subsystems (subvehicles). Each primary vehicle may also have any number of defined optical systems with each optical system having its own coordinate system and two-dimensional FOV. Each subvehicle also has a defined coordinate system and a three-dimensional shape.

Within the optical FOV, the object is located by an angular position with projected dimensions. The position of the observed vehicle or subvehicle is also computed in the Local Vertical-Local Horizontal (LVLH) system.

MATH MODEL

The RAP is a flight software utility routine which can be either called or scheduled in the Space Shuttle Functional Simulator (SSFS) program. The routine uses data which are generated by the active vehicle models GRAV2 and POSSUM. Currently, RAP is configured for three vehicles although it could be extended for N vehicles and M subvehicles. RAP requires the following user input:

NPV — Designated primary vehicle with values of 1 for the Space Shuttle Orbiter (SSO), 2 for the MTV, and 3 for another designated vehicle.

NSV — Designated secondary vehicle with the same description as NPV.

NSUBP — The designated primary optical system located on the primary vehicle. For the SSO, the values are 2 for the rear window, 3 for the overhead window, and 4 for the Flight Support Station (FSS). For the MTV, the values are 2 for the front camera and 3 for the rear camera.

NSUBS — The designated viewed system on the secondary vehicle. For the SSO, the values are 1 for the SSO vehicle, 2 for the rear window, 3 for the overhead window, and 4 for the FSS. For the MTV, the values are 1 for the MTV body, 2 for the front camera, 3 for the rear camera.

For example, if NPV = 1, NSV = 2, NSUBP = 2, and NSUBS = 1, then the Shuttle rear window is the designated optical system and the MTV body is the viewed object. The block data associated with the designated input parameters is presented in Appendix A.

After the systems have been defined, the position vector from the primary to secondary system in Earth-Centered Inertial (ECI) coordinates is computed.

$$\overline{RVTVI} = \overline{RTV} - \overline{RV} \qquad (1)$$

where

$\overline{RTV}$ is the ECI position vector of primary vehicle

$\overline{RV}$ is the ECI position vector for the secondary vehicle

The resultant vector, $\overline{RVTVI}$, is transformed from ECI to the primary vehicle system. This transformation and other transformations can be easily understood by referring to figure 3-8, which relates primary and secondary coordinate transformations. The vector $\overline{RVTV}$, which is the primary-to-secondary system vector expressed in the primary system, is computed by

$$\overline{RVTV} = (\overset{*}{TIP})\ (\overline{RVTVI})$$

where $\overset{*}{TIP}$ is the inertial-to-primary transformation matrix.

This relative position vector is adjusted for the optical offset and transformed to the optical system by

$$\overline{RVTVP} = (\overset{*}{TPO})(\overline{RVTV} - \overline{RVOFF}) \qquad (2)$$

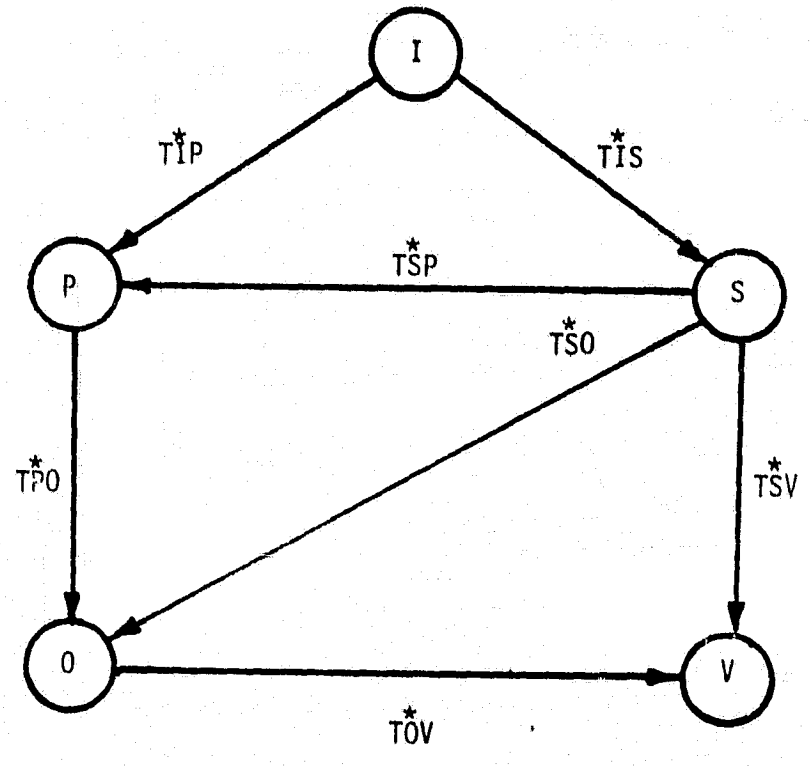
where

$\overline{RVOFF}$ is the offset vector of the optical system as defined in the primary system

$\overset{*}{TPO}$ is the primary to optical system transformation

Any offsets from the secondary to viewed system are computed with the final relative position vector in the optical system given by

$$\overline{RVTVP} = \overline{RVTVP} + (\overset{*}{TSO})\ (\overline{RTVOFF}) \qquad (3)$$



Coordinate system mnemonics:

I - Earth-Centered Inertial

P - Primary system

S - Secondary system

O - Optical system

V - Viewer system

Figure 3-8.- RAP coordinate systems and transformations.

where

$\overline{RTVOFF}$ is the viewed system offset defining the location of the designated subsystem (referred to the secondary coordinate system)

$\overset{*}{TSO}$ is the secondary to optical system transformation matrix

The vector $\overline{RVTVP}$ defines the Line-of-Sight (LOS) vector from the optical to viewed systems (see figs. 3-9 and 3-10). The magnitude of the LOS vector is

$$VMAG = \sqrt{(RVTVP(1)^2 + RVTVP(2)^2 + RVTVP(3)^2} \quad (4)$$

The dimensions of the viewed object are transformed to the optical system with the apparent dimensions in the optical FOV Y and Z axes computed as

$$Y_{ext} = |(X_{dim}/2)(TOV(2,1))| + |(Y_{dim}/2)(TOV(2,2))| + |(Z_{dim}/2)(TOV(2,3))| \quad (5)$$

$$Z_{ext} = |(X_{dim}/2)(TVO(3,1))| + |(Y_{dim}/2)(TOV(3,2))| + |(Z_{dim}/2)(TOV(3,3))| \quad (6)$$

where

$\overset{*}{TVO}$ is the viewed-to-optical transformation matrix

Xdim, Ydim, and Zdim are the viewed vehicle dimensions in the viewed system

The viewed vehicle apparent angular size as observed in the optical FOV is

$$AXS = \left(\frac{X_{dim}}{|\overline{RVEC}|}\right)\left(\hat{X}_B \times \hat{RVEC}\right)$$

$$AYS = \left(\frac{Y_{dim}}{|\overline{RVEC}|}\right)\left(\hat{Y}_B \times \hat{RVEC}\right) \quad (7)$$

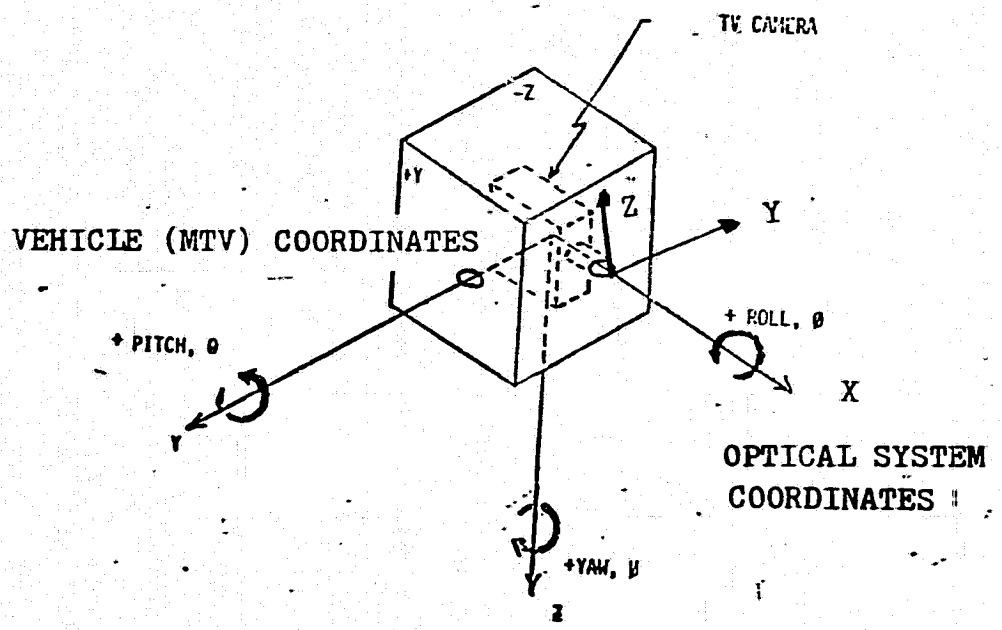
$$AZS = \left(\frac{Z_{dim}}{\overline{RVEC}}\right)\left(\hat{Z}_B \times \hat{RVEC}\right)$$

where

AXS, AYS, and AZS are the apparent angular dimensions of the viewed vehicle

$\overline{RVEC}$ is the vector from the viewed to optical system

$\hat{X}_B$, $\hat{Y}_B$, $\hat{Z}_B$ are defined viewed system unit axes



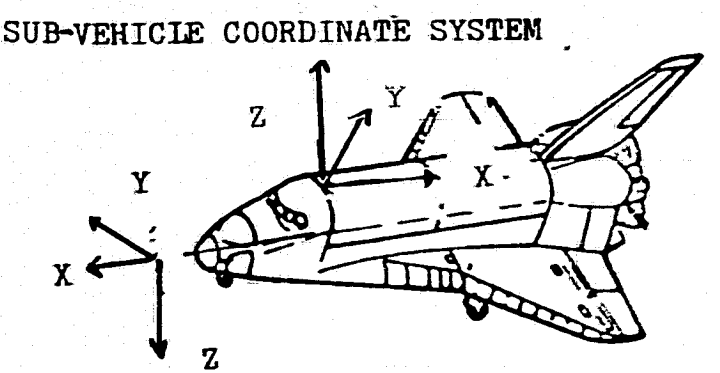


Figure 3-9.- Relative position and attitude processor coordinate systems.

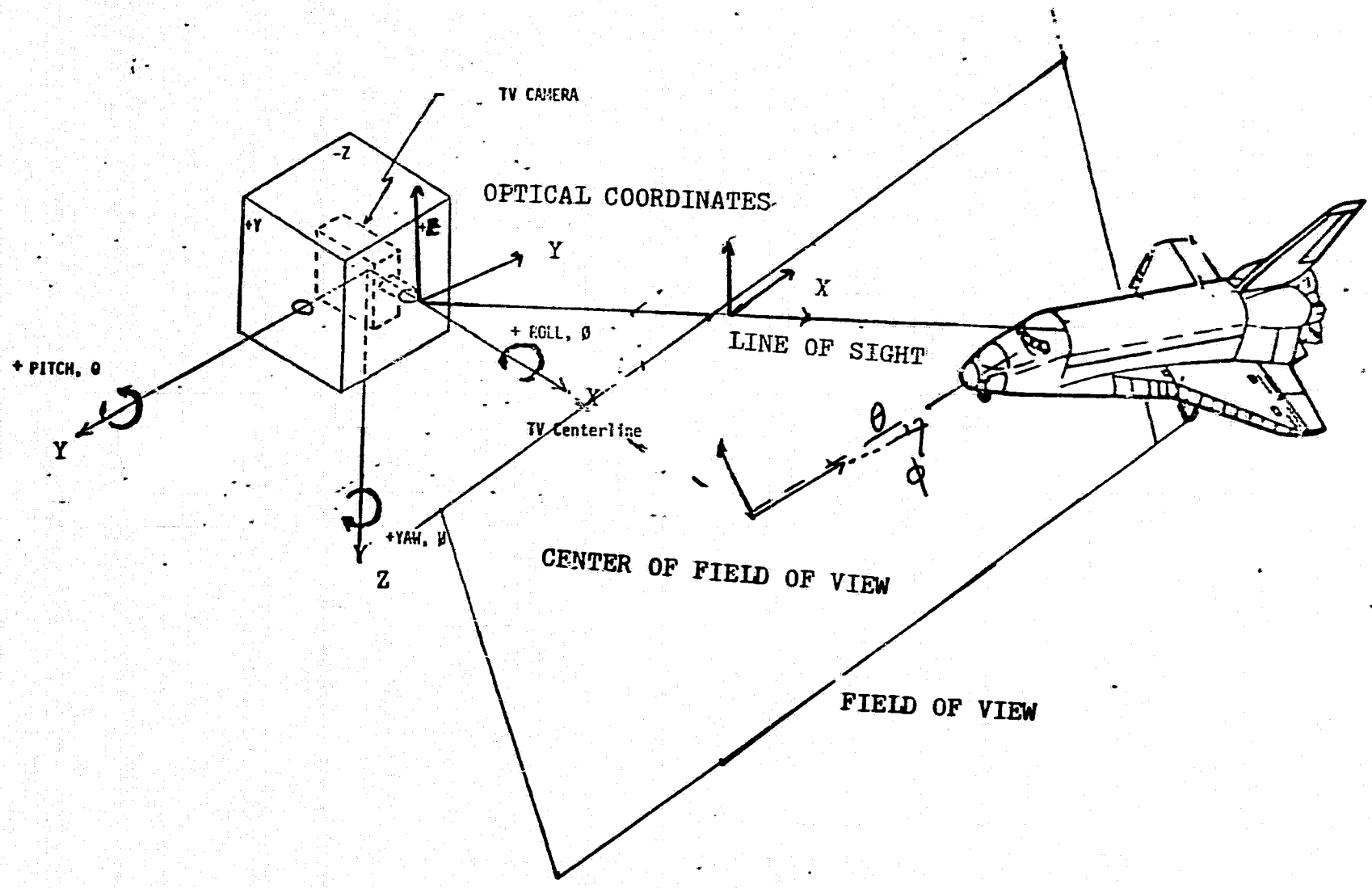


Figure 3-10.- Coordinates of subvehicle in optical FOV.

The position of the center of the viewed vehicle in the optical system defined as the horizontal (Y-axis) and vertical (Z-axis) angles is

$$\begin{aligned} HANG &= \tan^{-1}(RVTVP(2)/RVTVP(1)) \\ VANG &= \tan^{-1}(RVTVP(3)/RVTVP(1)) \end{aligned} \tag{8}$$

where $\overline{RVTVP}$ is the LOS vector in the optical system.

To determine whether the viewed object is within the FOV, the following computation is made:

$$\begin{aligned} HANG2 &= |HANG| - Y_{ext}/|\overline{RVTVP}| \\ VANG2 &= |VANG| - Z_{ext}/|\overline{RVTVP}| \end{aligned} \tag{9}$$

Computed angles HANG2 and VANG2 define the apparent angular distance from the viewed subvehicle to the closest point on the Y and Z optical axes. If both of these angles are less than the optical FOV dimensions, the viewed vehicle is within the FOV.

The routine RAP also determines whether the sun, moon, or earth is within the optical FOV by receiving ECI unit vectors describing the position of each object from the routine POSSUM which computes the current lunar and solar positions relative to ECI coordinates. POSSUM is called by the gravity environment model and updated as required by the integrator. These unit vectors are transformed to the optical system with the horizontal and vertical angles determined as previously discussed, except that the LOS vectors are replaced by the earth, sun, and moon unit vectors.

The Y_{ext} and Z_{ext} for the earth, sun, and moon are simply the apparent angular sizes of these objects. With this information, the FOV check is made and the occulation of the sun or moon by the earth is considered.

The position of the viewed object in the LVLH system (fig. 3-11) is computed using the vector $\overline{RLVLH}$, the vector from the Shuttle to the viewed object in the LVLH coordinate frame. The LVLH angles as defined are

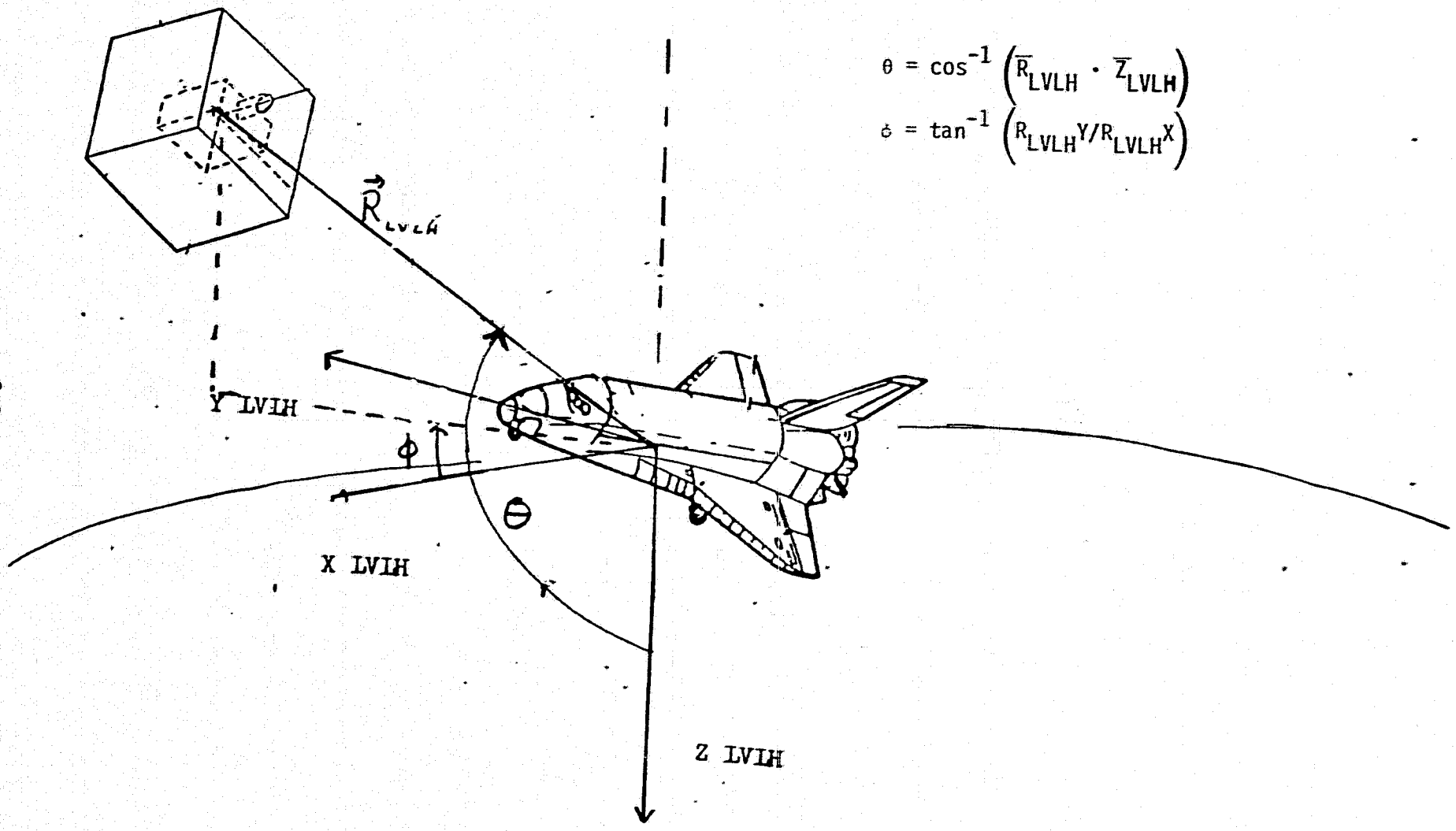


Figure 3-11.- LVLH coordinates.

$$\theta = \cos^{-1} (\overline{RLVLH} \cdot \hat{Z}LV)$$

$$\phi = \tan^{-1} (R_{LVLH}{}^{Y}/R_{LVLH}{}^{X})$$

where $\hat{Z}LV$ is the LVLH unit Z-axis defined by

$$\hat{Z}LV = -\hat{R}_{SSO}$$

where $\hat{R}_{SSO}$ is the unit ECI position vector of the Shuttle.

The relative position and velocity of the Shuttle and the viewed secondary object are given by

$$\begin{aligned} \overline{RELP} &= \overline{R}_{SSO} - \overline{R}_{obj} \\ \overline{VREL} &= \overline{V}_{SSO} - \overline{V}_{obj} \end{aligned} \tag{10}$$

where

$\overline{R}_{obj}$ is the viewed object ECI position vector

$\overline{V}_{SSO}$ is the ECI velocity of the Shuttle

$\overline{V}_{obj}$ is the ECI velocity of the viewed object, respectively.

RAP OUTPUT

The following variables output from RAP are defined as follows:

HANG2	Leading edge of object if within FOV; horizontal angle in degrees
HANG	Center of object in FOV; horizontal angle in degrees
VANG2	Leading edge of object if within FOV; vertical angle in degrees
VANG	Center of object in FOV; vertical angle in degrees
HANGSM(1)	Leading edge of sun; horizontal angle in degrees
VANGSM(1)	Leading edge of sun; vertical angle in degrees
HANGSM(2)	Leading edge of moon; horizontal angle in degrees
VANGSM(2)	Leading edge of moon; vertical angle in degrees
HANGSM(3)	Leading edge of earth; horizontal angle in degrees
VANGSM(3)	Leading edge of earth; vertical angle in degrees
PHILV, THETLV	Local vertical position angles of viewed vehicle in degrees

AXS AYS AZS	Apparent angular size of viewed object in degrees
RELP	ECI relative position vector between viewed and primary vehicles
VREL	ECI relative velocity vector between viewed and primary vehicles
RVTVP	Primary system relative position vector between viewed and primary vehicle

RAP

APPENDIX A

VEHICLE AND SUBVEHICLE DATA

```
      BLOCK DATA
C
      COMMON/FSIC/FS(1)
C
      DIMENSION DAT1(17,4,3)
      EQUIVALENCE (FS(150),DAT1(1,1,1))
      EQUIVALENCE (FS(100),NSUBP),(FS(101),NSUBS)
      EQUIVALENCE (FS(126),NPV),(FS(127),NSV)
C
      DATA NPV /0/
      DATA NSV /1/
      DATA NSUBP /2/
      DATA NSUBS /1/
C
C     LOCATION OF ORIGIN OF SSO IN SSO FRAME
C
      DATA (DAT1(I,1,1),I=1,3) /0.,0.,0./
C
C     MATRIX SSO BODY TO SS1(SSO)
C
      DATA DAT1(4,1,1) /1.0/
      DATA DAT1(8,1,1) /1.0/
      DATA DAT1(12,1,1) /1.0/
C
C     DIMENSIONS OF SSO IN SS1(SSO)
C
      DATA (DAT1(I,1,1),I=13,15) /144.7,97.06,25.52/
C
C     LOCATION OF ORIGIN OF SSO SUBSYSTEM 2 REL TO SSO
C
      DATA DAT1(1,2,1) /560. /
      DATA DAT1(2,2,1) /-.85/
      DATA DAT1(3,2,1) /-117./
C
C     TRANS SSO BODY TO SSO SUBSYSTEM 2(REAR WINDOW)
C
      DATA DAT1(4,2,1) /-1.0/
      DATA DAT1(8,2,1) /1.0/
      DATA DAT1(12,2,1) /-1.0/
C
C     DIMENSIONS OF SUBSYSTEM 2 IN THAT FRAME
C
      DATA DAT1(13,2,1),DAT1(14,2,1),DAT1(15,2,1)/1.,0.,1./
C
C     FOV OF SUBSYSTEM 2 SYSTEM(REAR WINDOW)
C
      DATA DAT1(16,2,1),DAT1(17,2,1)/31.,33./
C
C     LOCATION OF ORIGIN OF SSO(OVERHEAD WINDOW) SUBSYSTEM 3 REL TO SSO
C
      DATA DAT1(1,3,1) /578.2 /
      DATA DAT1(2,3,1) /-.85/
      DATA DAT1(3,3,1) /-122.4/
C
C     TRANS SSO BODY TO SSO SUBSYSTEM 2(OVERHEAD WINDOW)
```

```
C
      DATA DAT1(6,3,1) /1.0/
      DATA DAT1(8,3,1) /1.0/
      DATA DAT1(10,3,1) /-1.0/
C
C     DIMENSIONS OF SUBSYSTEM 3 IN THAT FRAME
C
      DATA DAT1(13,3,1),DAT1(14,3,1),DAT1(15,3,1)/1.,0.,1./
C
C     FOV OF SUBSYSTEM 3 SYSTEM(OVERHEAD WINDOW)
C
      DATA DAT1(16,3,1),DAT1(17,3,1)/42.,40./
C
C     LOCATION OF ORIGIN OF SSO SUBSYSTEM 4 REL TO SSO
C     FLIGHT SUPPORT STATION
C
      DATA DAT1(1,4,1) /-3.4 /
      DATA DAT1(2,4,1) /0.0/
      DATA DAT1(3,4,1) /-41.8/
C
C     TRANS SSO BODY TO SSO SUBSYSTEM 4(FSS)
C
      DATA DAT1(6,4,1) /1.0/
      DATA DAT1(8,4,1) /1.0/
      DATA DAT1(10,4,1) /-1.0/
C
C     DIMENSIONS OF SUBSYSTEM 4 IN THAT FRAME
C
      DATA DAT1(13,4,1),DAT1(14,4,1),DAT1(15,4,1)/9.,15.,4./
C
C     LOCATION OF ORIGIN OF MTV SUBSYSTEM 1 REL TO MTV
C
      DATA DAT1(1,1,2),DAT1(2,1,2),DAT1(3,1,2)/0.,0.,0./
C
C     TRANS MTV BODY TO MTV SUBSYSTEM 1
C
      DATA DAT1(4,1,2) /1.0/
      DATA DAT1(6,1,2) /1.0/
      DATA DAT1(10,1,2) /1.0/
C
C     DIMENSIONS OF MTV IN MTV SYSTEM
C
      DATA DAT1(13,1,2),DAT1(14,1,2),DAT1(15,1,2)/3.5,2.58,2.58/
C
C     LOCATION OF ORIGIN OF MTV SS2 IN SS1 FRAME(FORWARD CAMERA)
C
      DATA (DAT1(I,2,2),I=1,3) /2.,0.,0./
C
C     MATRIX SS1 TO SS2 FRAME
C
      DATA DAT1(4,2,2) /1.0/
      DATA DAT1(8,2,2) /1.0/
      DATA DAT1(12,2,2) /1.0/
C     DIMENSIONS OF SUBSYSTEM 2 IN 2 FRAME
C
      DATA (DAT1(I,2,2),I=13,15) /0.,0.,0./
```

```
C
C     FOV OF MTV FORWARD CAMERA
      DATA DAT1(16,2,2),DAT1(17,2,2)/40.,40./
C     LOCATION OF ORIGIN OF MTV SS3 IN SS1 FRAME(REAR CAMERA)
      DATA (DAT1(I,3,2),I=1,3) /-2.,0.,0./
C     MATRIX SS1 TO SS3 FRAME
      DATA DAT1(4,3,2) /-1.0/
      DATA DAT1(8,3,2) /1.0/
      DATA DAT1(12,3,2) /-1.0/
C     DIMENSIONS OF SUBSYSTEM 3 IN 3 FRAME
      DATA (DAT1(I,3,2),I=13,15) /0.,0.,0./
C     FOV OF MTV REAR CAMERA
      DATA DAT1(16,3,2),DAT1(17,3,2)/16.,16./
C     LOCATION OF ORIGIN OF MTV2 SUBSYSTEM 1 REL TO MTV2
      DATA DAT1(1,1,3),DAT1(2,1,3),DAT1(3,1,3)/0.,0.,0./
C
C     TRANS MTV2 BODY TO MTV2 SUBSYSTEM 1
      DATA DAT1(4,1,3) /1.0/
      DATA DAT1(8,1,3) /1.0/
      DATA DAT1(12,1,3) /1.0/
C     DIMENSIONS OF MTV2 IN MTV2 SYSTEM
      DATA DAT1(13,1,3),DAT1(14,1,3),DAT1(15,1,3)/3.5,2.58,2.58/
C     LOCATION OF ORIGIN OF MTV2 SS2 IN SS1 FRAME(FORWARD CAMERA)
      DATA (DAT1(I,2,3),I=1,3) /2.,0.,0./
C     MATRIX SS1 TO SS2 FRAME
      DATA DAT1(4,2,3) /1.0/
      DATA DAT1(8,2,3) /1.0/
      DATA DAT1(12,2,3) /1.0/
C     DIMENSIONS OF SUBSYSTEM 2 IN 2 FRAME
      DATA (DAT1(I,2,3),I=13,15) /0.,0.,0./
C     FOV OF MTV2 FORWARD CAMERA
      DATA DAT1(16,2,3),DAT1(17,2,3)/40.,40./
C     LOCATION OF ORIGIN OF MTV2 SS3 IN SS1 FRAME(REAR CAMERA)
      DATA (DAT1(I,3,3),I=1,3) /-2.,0.,0./
C
```

```
C       MATRIX SS1 TO SS3 FRAME
C
        DATA DAT1(4,3,3) /-1.0/
        DATA DAT1(8,3,3) /1.0/
        DATA DAT1(12,3,3) /-1.0/
C       DIMENSIONS OF SUBSYSTEM 3 IN 3 FRAME
C
        DATA (DAT1(I,3,3),I=13,15) /0.,0.,0./
C
C       FOV OF MTV2 REAR CAMERA
C
        DATA DAT1(16,3,3),DAT1(17,3,3)/15.,16./
        END
```

.POSSUM/SP

RAP

APPENDIX B

PROGRAM LISTING AND COMMON BLOCK LOCATIONS

```
EJ3=L75038*RUNS(1),RAP3
      1            SUBROUTINE RAP3
      2      C
      3      C     THIS ROUTINE DETERMINES THE PRIMARY LOS VECTOR
      4      C     AND EULER ANGLES RELATIVE PRIMARY OPTICAL IC LOS FRAME.
      5      C     A CHECK TO SEE IF THE VEHICLE,SUN,MOON IS
      6      C     IN FOV IS MADE.
      7      C
      8      C     NPVEI FOR SSC, 2 FOR NIV, 3...N FOR OTHERS (PRIMARY OPTICAL VEHICLE
      9      C     NSVEI FOR SSC, 2 FOR NIV, 3...N FOR OTHERS (VIEWED VEHICLE
     10      C     NSUNMO=1 IF SUN/MOON CHECK REQUIRED
     11      C     HANGSM(1,1) = SUN HORIZONTAL FOV ANGLE PRIMARY
     12      C     HANGSM(2,1) = SUN HORIZONTAL FOV ANGLE SECONDARY
     13      C     HANGSM(1,2) = MOON PRIMARY HOR ANGLE
     14      C     HANGSM(2,2) = MOON SEC HOR ANGLE
     15      C     VANGSM = SAME AS ABOVE FOR VERTICAL ANGLE
     16      C
     17            REAL MTF
     18      C
     19            COMMON /ACTVEC/AVC(1)
     20            COMMON /ACTIVC/AVC1(1)
     21            COMMON /ACTV2C/AVC2(1)
     22            COMMON /CONST/CC(1)
     23            COMMON /FSTC/FS(1)
     24            COMMON /GRAVC/G(1)
     25      C
     26            DIMENSION RV(3),RTV(3),TIP(9),TIS(9),TPO(9),TVO(9),
     27           *          RVOFF(3),RTVOFF(3),TPI(9),TSI(9),SC(30),
     28           *          TPS(9),TSP(9),RVIVP(3),XOP(3),RLV(3),
     29           *          YOP(3),ZOP(3),RVTV(3),RELV(3),VELV(3)
     30            DIMENSION  ZLV(3),YLV(3),XLV(3),RUNIT(3),TLV(9),RELP(3)
     31            DIMENSION VREL(3),XBOD(3),YBOD(3),ZBOD(3),RVTVI(3),T(9)
     32            DIMENSION ZBODM(3),XLOS(3),YLOS(3),ZLOS(3),DAIL(17,4,3)
     33            DIMENSION HANGSM(2,3),VANGSM(2,3),LSMFLG(2,3),JV(3,3),APDIAM(3)
     34      C
     35            EQUIVALENCE (CC(34),MTF),(CC(31),GTR)
     36            EQUIVALENCE (CC(26),EMU),(CC(32),RTD)
     37            EQUIVALENCE (FS(100),NSUBP),(FS(101),NSUBS),(FS(105),RELP)
     38            EQUIVALENCE (FS(109),VREL(1)),(FS(112),THETLV)
     39            EQUIVALENCE (FS(113),PHILV),(FS(114),THETLK)
     40            EQUIVALENCE (FS(115),PHILK),(FS(116),PHI)
     41            EQUIVALENCE (FS(100),LAT1(1,1,1)),(FS(103),THETA)
     42            EQUIVALENCE (FS(104),HANG),(FS(105),LFLAG)
     43            EQUIVALENCE (FS(117),RVTVP(1)),(FS(120),AXS)
     44            EQUIVALENCE (FS(121),AYS),(FS(122),AZS)
     45            EQUIVALENCE (FS(123),YEXT),(FS(124),ZEXT),(FS(125),VANG)
     46            EQUIVALENCE (FS(126),NPV),(FS(127),NSV)
     47            EQUIVALENCE (FS(129),HANGSM),(FS(135),VANGSM)
     48            EQUIVALENCE (FS(141),NSUNMO),(FS(142),IPFLG)
     49            EQUIVALENCE (G(78),UV(1,1)),(G(102),APDIAM(1))
     50      C
     51            DATA XLP /1.,0.,0./
     52            DATA YLP /0.,1.,0./
     53            DATA ZLP /0.,0.,1./
     54            DATA ZLODM/0.,0.,-1./
     55            DATA XLOS /1.,0.,0./
     56            DATA YLOS /0.,1.,0./
```

```
      DATA Z_OD /0.,0.,1./
C
      NAMELI _T/NAME1/HANG,VANG,HANG2,VANG2,RVTVP,YEXT,ZEXT,
     *VMAG,T FO,TVG,RVOFF,RTVOFF,XLOS,YLOS,ZLOS
C
C
C
      IPASS=_
C
C     DETERM NE IF SSC IS PRIMARY AND MTV IS VIEWED (OR REVERSE)
    1 IF(N.PV .EC._.AND.N.CV.EC.2)GO TO 10
      IF(N.PV .EC._.AND.N.CV.EC.1)GO TO 20
      IF(N.PV .EC.1.AND.N.CV.EC.3)GO TO 23
      IF(N.PV .EC.2.AND.N.CV.EC.3)GO TO 26
   10 DO 15 I=1,3
      RV(I)=AVC(I)*MTF
      RELV(I )=AVC1(I)
      VELV(I )=AVC1(I+3)
   15 RTV(I) =AVC1(I)*MTF
      CALL U NIT(AVC(1),LV(1,3))
      RMAG=A _VAL(AVC(1))
      X=EN.R/ RMAG
      APOLAM (3)=2.0 * ASIN(X)*RTD
      DO 16 I=1,9
C
C     TIP  =  TRANS INERTIAL TO PRIMARY
C     TSI  =  TRANS SECONDARY TO INERTIAL
C     TPI  =  TRANS PRIMARY TO INERTIAL
C     TIS  =  TRANS INERTIAL TO SECONDARY
C
      TIP(I) =AVC(I+99)
      TPI(I) =AVC(I+36)
      TSI(I) =AVC1(I+36)
   16 TIS(I) =AVC1(I+99)
      GO TO 30
   20 DO 21 I=1,3
      RV(I)= AVC1(I)*MTF
      REL (I )=AVC1(I)
      VELV(I )=AVC1(I+3)
   21 RTV(I) =AVC(I)*MTF
      CALL U NIT(AVC1(1),LV(1,3))
      RMAG=A _VAL(AVC1(1))
      X=EN.R/ RMAG
      APOLAM (3)=2.00*ASIN(X)*RTD
      DO 22 I=1,9
      TPI(I) =AVC1(I+38)
      TSI(I) =AVC(I+38)
      TIP(I) =AVC1(I+99)
   22 TIS(I) =AVC(I+99)
      GO TO 30
   23 DO 24 I=1,3
      RV(I)= AVC(I)*MTF
      RELV(I )=AVC2(I)
      VELV(I )=AVC2(I+3)
   24 RTV(I) =AVC2(I)*MTF
      CALL U NIT(AVC(1),LV(1,3))
      RMAG=A _VAL(AVC(1))
```

```
      X=ERR/RMAG
      APDIAM(3)=2.0 * ASIN(X)*RTD
      DO 25 I=1,9
      TIP(I)=AVC(I+99)
      TPI(I)=AVC(I+38)
      TSI(I)=AVC2(I+38)
   25 TIS(I)=AVC2(I+99)
      GO TO 30
   26 DO 27 I=1,3
      RV(I)=AVC1(I)*MTF
      RELV(I)=AVC2(I)
      VEL(I)=AVC2(I+3)
   27 RTV(I)=AVC2(I)*MTF
      CALL UNIT(AVC1(1),UV(1,3))
      RMAG=MAGVAL(AVC1(1))
      X=ERR/RMAG
      APDIAM(3)=2.00*ASIN(X)*RTD
      DO 28 I=1,9
      TPI(I)=AVC2(I+38)
      TSI(I)=AVC2(I+38)
      TIP(I)=AVC2(I+99)
   28 TIS(I)=AVC2(I+99)
   30 CONTINUE
      DO 31 I=1,3
      RVOFF(I)=DAT1(I,NSUBP,NPV)/12.
   31 RTVOFF(I)=DAT1(I,NSUBS,NSV)/12.
      DO 32 I=1,9
C
C     TPO = TRANSF PRIMARY SYSTEM TO PRIMARY OPTICAL SYSTEM
C     TVO = TRANSF SECONDARY SYSTEM TO OFFSET SYSTEM
C
      TPO(I)=DAT1(I+3,NSUBP,NPV)
   32 TVO(I)=DAT1(I+3,NSUBS,NSV)
      XVEC=DAT1(13,NSUBS,NSV)
      YVEC=DAT1(14,NSUBS,NSV)
      ZVEC=DAT1(15,NSUBS,NSV)
      FOV1=DAT1(16,NSUBP,NPV)
      FOV2=DAT1(17,NSUBP,NPV)
C
C     COMPUTE POS VECTOR PRIMARY TO VIEWED
C
      DO 35 I=1,3
   35 RVTVI(I)=RTV(I)-RV(I)
C
C     TRANSFORM TO PRIMARY FRAME
C
      CALL MXV(TIP,RVTVI,RVTV)
C
C     COMPUTE PRIMARY TO SECONDARY MATRIX PLUS XPOSE
C
      CALL MXM(TIS,TPI,TPS)
      CALL XPOSE(TPS,TSP)
C
C     ADJUST FOR OPTICAL AND VIEWED OFFSETS IF ANY
C
      DO 40 I=1,3
   40 SC(I)=RVTV(I)-RVOFF(I)
```

```
C
C     TRANSFORM TO OPTICAL SYSTEM, RVTVP IN OPTICAL
C
      CALL MXV(TPC,SC(1),RVTVP(1))
C
C     COMPUTE VIEWED OFFSET TO PRIMARY OPTICAL MATRIX
C
C     T = SECONDARY TO PRIMARY OFFSET TRANSFORMATION
C
      CALL MXM(TPC,TSP,T(1))
C
      CALL MXV(T(1),RTVOFF,SC(1))
      DO 45 I=1,3
   45 RVTVP(I)=RVTVP(I)+SC(I)
C
C
C     CONSTRUCT LOS FRAME WITH ORIGIN AT PRIMARY OFFSET ORIGIN
C
      CALL UNIT(RVTVP,XLOS)
      CALL CROSS(XLOS,YCP,SC(1))
      CALL UNIT(SC(1),ZLOS)
      CALL CROSS(ZLOS(1),XLOS(1),SC(1))
      CALL UNIT(SC(1),YLOS(1))
C
C     DETERMINE IF VIEWED VEH IS IN PRIMARY FOV
C
C     COMPUTE ANGLE BETWEEN PRIMARY OPTICAL AXIS  AND ACTUAL LOS VECTOR
C
      THETA=ACOS(XLOS(1))
      PHI=ATAN2(RVTVP(3),RVTVP(2))
C
      VMAG=SQRT(RVTVP(1)**2 + RVTVP(2)**2 + RVTVP(3)**2)
C
C     TRANSFORM DIMENSIONS OF VIEWED OBJECT INTO OPTICAL SYSTEM
C
      CALL XPOSE(TVO,SC(10))
      CALL MXM(SC(10),T(1),SC(1))
      YEXT=ABS(SC(2)*XV/2)+ABS(SC(5)*YV/2)
     C     +ABS(SC(8)*ZV/2)
      ZEXT=ABS(SC(3)*XV/2)+ABS(SC(6)*YV/2)
     C     +ABS(SC(9)*ZV/2)
C
C     COMPUTE PRIMARY OFFSET TO VIEWED
C
      CALL XPOSE(T,SC(1))
      CALL MXM(SC(1),TVO(1),SC(20))
C
C     TRANSFORM RVTVP BACK TO VIEWED SYSTEM
C
      CALL MXV(SC(20),RVTVP(1),SC(13))
      CALL UNIT(SC(10),SC(1))
C     COMPUTE APPARENT ANGULAR SIZE
C
      CALL CROSS(XBOD(1),SC(1),SC(4))
      SC(7)=ABVAL(SC(4))
      AXS=XV*VMAG*SC(7)
      CALL CROSS(YBOD(1),SC(1),SC(4))
```

```
      SC(7)=ABVAL(SC(4))
      AYS=YV*VMAG*SC(7)
      CALL CROSS(ZBOD(1),SC(2),SC(4))
      SC(7)=ABVAL(SC(4))
      AZS=ZV*VMAG*SC(7)
C
C     DETERMINE IF VIEWED IN PRIMARY REC FOV
C
      HANG=ATAN2(RVTVP(2),RVTVP(1))
      VANG=ATAN2(RVTVP(3),RVTVP(1))
      HANG2=ABS(HANG)-YCXT/VMAG
      VANG2=ABS(VANG)-ZCXT/VMAG
C
      LFLAG=0
      IF((HANG2).LT.FOV1*DTR/2.AND.(VANG2).LT.FOV2*DTR/2)LFLAG=1
      IF(LFLAG.EQ.1)WRITE(6,152)NSV,NPV
      IF(LFLAG.EQ.2)WRITE(6,162)NSV,NPV
  152 FORMAT(/3CX,I2,2X,'IN FOV OF',2X,I2)
  162 FORMAT(/3CX,I2,2X,'NOT IN FOV OF',2X,I2)
C
C     SUN/MOON-EARTH CHECKS
      J=0
      IF(I.SUNMO.EQ.0)GO TO 220
  100 J=J+1
C
      IF(J.EQ.3)GO TO 181
      CALL MXM(TPO,TIP,SC(10))
      CALL MXV(SC(10),DV(1,J),SC(1))
      CALL MXV(SC(10),DV(1,3),SC(20))
      ANG=DOT(SC(1),SC(20))
      EANG=ACOS(ANG)*RTD
      IF(EANG.LT.APDIAM(3)*0.5)GO TO 219
  101 CONTINUE
      DO 185 I=1,3
  185 SC(I)=DV(I,J)
      CALL MXM(TPO(1),TIP,SC(10))
      CALL MXV(SC(10),SC(1),SC(7))
      APD=APDIAM(J)*DTR*.5
      CALL UNIT(SC(7),SC(4))
      HANGSM(NPV,J)=ABS(ATAN2(SC(5),SC(4)))
      VANGSM(NPV,J)=ABS(ATAN2(SC(6),SC(4)))
      LSMFLG(NPV,J)=2
      IF((HANGSM(NPV,J)-APD).LT.FOV1*DTR/2.AND.
     *(VANGSM(NPV,J)-APD).LT.FOV2*DTR/2)LSMFLG(NPV,J)=1
      IF(LSMFLG(NPV,J).EQ.1.AND.J.EQ.1)WRITE(6,216)NPV
      IF(LSMFLG(NPV,J).EQ.1.AND.J.EQ.2)WRITE(6,217)NPV
      IF(LSMFLG(NPV,J).EQ.1.AND.J.EQ.3)WRITE(6,218)NPV
  218 FORMAT(/3CX,'EARTH IN FOV OF',2X,I2)
  216 FORMAT(/3CX,'SUN IN FOV OF',2X,I2)
  217 FORMAT(/3CX,'MOON IN FOV OF',2X,I2)
  219 CONTINUE
      IF(J.LT.3)GO TO 100
  220 CONTINUE
C     CONSTRUCT LCLV SYSTEM
C
      CALL UNIT(AVC(1),ZLV(1))
      DO 300 I=1,3
```

```
      300  ZLV(I)=ZLV(I)
           CALL CROSS (AVC(4),AVC(I),SC(I))
           CALL UNIT(SC(1),YLV(1))
           CALL CROSS(YLV(1),ZLV(1),SC(1))
           CALL UNIT(SC(1),XLV(1))
      C
      C    CONSTRUCT TRANSFORMATION M50 TO ZLV
      C
           TLV(1)=XLV(1)
           TLV(2)=YLV(1)
           TLV(3)=ZLV(1)
           TLV(4)=XLV(2)
           TLV(5)=YLV(2)
           TLV(6)=ZLV(2)
           TLV(7)=XLV(3)
           TLV(8)=YLV(3)
           TLV(9)=ZLV(3)
      C
      C    TRANSFORM LOS VECTOR FROM M50 TO LV
      C
           DO 325 I=1,3
      325  SC(I)=AVC1(I)-AVC(I)
           CALL MXV(TLV(I),SC(1),SC(4))
           CALL UNIT(SC(4),RLV(1))
      C
      C    COMPUTE SPACE ANGLES OF VIEWED VEHICLE IN LV SYSTEM
      C
           DTT=DOT(RLV(1),ZLV(1))
           THETLV=ACOS(DTT)
           PHILV=ATAN2(SC(2),SC(3))
      C
      C    COMPUTE LOOK ANGLES OF VIEWED VEHICLE IN BODY SYSTEM
      C
           DO 390 I=1,3
      390  RELP(I)=(RELV(I)-AVC(1))*MTF
           CALL MXV(AVC(100),RELP(1),SC(4))
           CALL UNIT(SC(4),RLNIT(1))
           DTT=DOT(RLNIT(1),ZBODM(1))
           THETLK=ACOS(DTT)
           PHILK=ATAN2(RELP(2),RELP(3))
      C    NOTE POSSIBLE PROBLEM WITH ATAN2
      C
      C    COMPUTE RELATIVE VELOCITIES OF SSO AND TARGET IN M50
      C
           DO 450 I=1,3
           RELP(I)=(AVC(I)-RELV(I))*MTF
      450  VREL(I)=(AVC(I+3)-VELV(I))*MTF
      C
           CALL FREE(
           RETURN
           END

@PRT,S RUNS.RAP3DP
```

SGN

Function SGN (WORD)

SGN returns a 1 if WORD is positive, a 0 if WORD is 0, and a -1 if WORD is negative.

SIG

Function SIG (WORD1, WORD2, NSDIG)

SIG returns a 0 if WORD1 is equal to WORD2 to NSDIG significant digits, a 1 if WORD1 is greater than WORD2 in NSDIG significant digits, and a -1 otherwise.

SIGNM

Function SIGNM (WORD)

SIGNM returns a 1 if WORD is positive or 0, and a -1 if WORD is negative.

3.5 PLOTTER ROUTINES

These routines process data from a SOAP simulation and cause plots to be generated. In general, they read and rearrange the data stored on logical unit 20 (and possibly 29) and create the calls to plotting utility routines which will do the actual point-by-point plotting. The plots which can be generated from a given simulation can include only that data specified upon preprocessing. Other information will not be saved in the appropriate form. Typically, the user will neither modify nor directly call these routines. It is only necessary to supply the proper plotting instruction cards (see section 2) and to execute the available code.

Routines described in this section are:

AXIS	FSPLOT	NULINE	PLMAIN	PRPLOT	SCALIE	SYMBOL
ENDFLE	LINCNT	NUMBER	PLOT	QUIKML	SKPFLE	TREAD
FACTOR	MINMAX	PACK	PLOTS	SCALE	SPMAIN	WHERE

AXIS

Subroutine AXIS (XP, YP, IBCD, ±NCHAR, AXLEN, ANG, FIRSTV, DELTAV)

AXIS is a CalComp subroutine which draws the graph axes, draws tic marks each inch, writes the value at each tic mark, and then labels the axes. AXIS calls NUMBER, PLOT, and SYMBOL.

XP, YP are the coordinates, in inches, of the axis starting point.

IBCD is the literal title.

±NCHAR defines number of characters and position of IBCD.

AXLEN is axis length in inches.

ANG is number of degrees from X axis.

FIRSTV is the starting value of first tic mark.

DELTAV is number of data units/inch of axis.

ENDFLE

Subroutine ENDFLE (NLU)

ENDFLE writes a coded end-of-file on a specified logical unit, NLU.

FACTOR

Subroutine FACTOR (FACT)

FACTOR is an entry point in the CalComp plot package which allows the user to scale up or down the entire plot. FACT is the scale factor applied to the plot. A value of 2.0 doubles the plot size. A value of 1.0 is the default value.

FSPLOT

Subroutine FSPLOT (NLU, NPLOTS, NWDR)

FSPLOT contains most of the logic of the production plotter. FSPLOT is called by PLMAIN where NLU is the input logical unit and NWDR is the number of plot variables. NPLOTS is a dummy argument, required in the argument list but currently not used. FSPLOT first skips to the correct data file. FSPLOT then reads in plot request cards until either 10 cards or 10 different parameters are obtained. These variables to be plotted are then read into core. A large loop is then started to produce the requested plots. PRPLOT is used for printer plots while QUIKML is used for SD-4060 plots. For CalComp plots, PLOTS, PLOT, SCALE (or SCALIE), AXIS, SYMBOL, and LINE are used. After all plot requests are processed, control is returned to PLMAIN.

LINCNT

Subroutine LINCNT sets the top and bottom page margins to nonstandard UNIVAC values for accommodating SOAP page plots.

MINMAX

Subroutine MINMAX (A1, K, AMIN, AMAX)

MINMAX defines the minimum, AMIN, and maximum, AMAX, of an array of K numbers beginning with A1.

NULINE

Subroutine NULINE (XA, YA, NPTS, INC, LN, IQ,T)

NULINE, a CalComp routine, is called to plot the actual data, one line or plot at a time, using subroutines WHERE, SYMBOL, and PLOT.

XA and YA are data arrays.

NPTS is number of points.

INC is frequency of plotted data.

LN describes type of line.

IQ is the CalComp integer equivalent of the desired symbol.

T is the array of angles by which symbols IQ are to be rotated.

NUMBER

Subroutine NUMBER (XP, YP, HT, FPN, ANG, NDEC)

NUMBER is a CalComp routine which translates numbers to BCD and writes them on a graph (using SYMBOL).

XP and YP are coordinates in inches from lower left corner.

HT is height in inches of the plotted number.

FPN is the floating point number

ANG is the angle from the X axis.

NDEC is number of decimal digits for output.

PACK

Subroutine PACK (WORD, CHAR, IPOS)

Pack is called by PRPLOT to pack the BCD word WORD, where CHAR indicates the direction of shift of IPOS characters.

PLMAIN

PLMAIN is the main program of the absolute element PLOTER used in execution of the production plotter. PLMAIN is a small driver program which uses FSPLOT to produce the desired plots. PLMAIN first reads a plotter control card to determine where the data is. If it is on tape, the data is transferred to a high speed drum unit to speed up the data reading. FSPLOT is then called for the actual plots. When control returns to PLMAIN, another control card is read if specified and the process described above is repeated. After all control cards are read, PLOT is called to terminate the CalComp plot tape.

PLOT

Subroutine PLOT (XP, YP, $\pm$IP)

PLOT is a CalComp routine which generates and writes instructions on tape to the CalComp hardware to produce the plots. Instructions to PLOT include (1) move the pen (with it down if IP = 2 or up if IP = 3) from the current position to the position specified by XP and YP; (2) end the plot if IP = -2 or -3 and move the pen to a new origin specified by XP and YP; (3) end the plot tape if IP = 999.

PLOTS

Subroutine PLOTS (WORK, NLOC, LU)

PLOTS is an entry point in the PLOT subroutine called for initialization of PLOT. WORK is the storage array name, NLOC is the size of the WORK array and LU is the logical unit on which the plot tape is to be mounted.

PRPLOT

Subroutine PRPLOT (AX, AY, K, IH, IC, ITITLE, IDM)

PRPLOT is called by SPMAIN and FSPLOT to produce printer plots.

AX is the array of X values.

AY is a two-dimensional array containing Y values for all curves in the frame.

K is the number of points in each curve.

IH is an array of characters for each curve.

IC is the number of curves in the plot.

ITITLE is a title array.

IDM is the number of points in AX.

QUIKML

Subroutine QUIKML (+L, XMIN, XMAX, YMIN, YMAX, ISYM, BCDX, BCDY, NP, X, Y).

QUIKML is a UNIVAC 1110 routine used to produce SD-4060 microfilm plots.

L is the number of grids to be drawn on one frame. If L is negative, the frame will be advanced.

XMIN, XMAX are minimum and maximum values of X array.

YMIN, YMAX are minimum and maximum values of Y array.

ISYM is the symbol to be used in plotting.

BCDX, BCDY are alphanumeric titles for the X and Y axes, respectively.

NP is the number of plot points.

X, Y are initial locations of X and Y arrays.

SCALE

Subroutine SCALE (X, S, N, K)

SCALE is a CalComp routine which defines a scale for a CalComp plot. SCALE defines the scale based on the desired plot size and the range of data such that is easy to interpolate between divisions on the resultant scale.

X is the data array to be scanned. An adjusted minimum value will be stored in X (N*K + 1). An adjusted DX (maximum - minimum) will be stored in X (N* K + K + 1)

S is the length over which data is to be plotted.

N is the number of data points in the X array.

K is the repeat cycle of a mixed array.

SCALIE

Subroutine SCALIE (X, S, N, K)

SCALIE is an alternate scale routine used with CalComp plots which produces larger plots than SCALE but less easy interpolation between divisions.

X(1) is minimum value to be plotted.
X(2) is maximum value to be plotted.
X(3) is the output adjusted minimum value.
X(4) is the adjusted delta value per inch.

S is the maximum length of plot in inches.

N and K are dummy arguments.

SKPFLE

Subroutine SKPFLE (NLU)

SKPFLE is used to skip a data file on a specified logical unit, NLU.

SPMAIN

SPMAIN is the main program of the absolute element SPDPLT used in execution of the high speed plotter. To reduce execution time, SPMAIN plots data from core instead of continually reading a data tape. To enable this, SPMAIN restricts data tapes to 30 variables with 1350 points per variable per file. SPMAIN ignores additional data or parameters if either limitation is exceeded. SPMAIN first reads a control card to find out where the data is and then brings the data into core. If CalComp plots have been specified, the maximum and minimum for each variable is determined, assuming that all will be plotted. A large loop is then entered to produce the requested plots. For CalComp plots, PLOTS, PLOT, SCALE (or SCALIE), AXIS, SYMBOL, and LINE are used. For printer plots, the data to be plotted is reduced to 100 points or less corresponding to the available page field width. PRPLOT is used for printer plots, if requested.

SYMBOL

Subroutine SYMBOL (X, Y, HT, IBCD, ANG, N)

SYMBOL is a CalComp routine which will plot N alphanumeric characters, IBCD, at a height of HT inches and angle of ANG degrees beginning at coordinates specified by X and Y.

TREAD

Subroutine TREAD (NLU, NW, MW, IP, IK, BUFF)

TREAD is used to read a data record, BUFF, from the input logical unit, NLU.

NW and MW are the first and last indices of the BUFF array.

IP is a parity indicator.

IK is a read operation indicator.

WHERE

Subroutine WHERE (X, Y, F)

WHERE is an entry point in the CalComp routine PLOT. WHERE is called by LINE to define the current CalComp pen position where X and Y are plot coordinates and F is a scale factor.

4. ENVIRONMENT MODELS

The Environment models serve to calculate the state vector (position, attitude, and rate of change of each) of each vehicle. They support the computation of forces and torques, the updating of mass properties, and the integration of the equation of motion. Environment models have a special position within the SOAP system since they allow input and output through individual COMMON blocks with names which are known to the SOAP system, since they perform their computations in a monolithic block so that all parameters are updated at once, and since model specifications (typically what number of vehicles) are made at the time of preprocessing and at the time of mapping. Typically, Environment models are not scheduled.

It should be noted that default values, which occur in lieu of specification of the Environment model at the preprocessing and mapping stages, will result in the use of a dummy model and the function of that model will be omitted.

4.1 ACTIVE VEHICLE

PROGRAM NAME

Single precision ACTVEH
Double precision AVEH/AVEH50

PROGRAM DESCRIPTION

ACTVEH is a driver which calls active vehicle models for each free flier. These models update the position and attitude of each vehicle. ACTVEH resets the flag IPDYN in ENVCOM prior to calling each active vehicle. This flag informs the vehicle models whether to update the dynamics derivatives (linear and angular accelerations), whether to update the active vehicle state, and in which order to perform these actions. ACTVEH updates the active vehicle internal times (AVTIM) to the update time required.

ACTVEH has no calling arguments and accepts no commands.

PROGRAM NAME

Single precision	AVES24 AVES19
Double precision	AVED19 AVED24 AVED242

The single- and double-precision versions of this model differ in the names of some of the subroutines called but are otherwise the same in operation and formulation.

PROGRAM DESCRIPTION

The active vehicle programs provide six-degree-of freedom simulations of a rigid body vehicle in three-dimensional inertial space. The vehicle dynamics use the forces and torques caused by firing jets of an Attitude Control

Propulsion System (ACPS), which is modeled by the Reaction Control System (RCS). The gravitational acceleration of earth (GRAV model), the torque due to the gravity gradient (GGT model) and the aerodynamic forces and torques (AERO model) are used in the vehicle dynamics. Computation of the vehicle's state vectors is performed relative to a nonrotating orthogonal coordinate system with the origin at the center of the earth (earth-centered inertial (ECI)). The rotational sequence used is roll, pitch, yaw. Each model is supported by a COMMON block (e.g., ACTVEC, ACTV1C, ACTV2C) and a block data routine which initializes the COMMON block.

MATH MODEL

The AVES24 program maintains the vehicle state vectors in ECI coordinates. The contact forces and torques are in the vehicle body frame. By summing these forces and torques, the contact acceleration of the vehicle is calculated.

$$\underline{F}_{AV} = \sum \underline{F}_{RCS} + \underline{F}_{AERO} \tag{1}$$

$$\underline{T}_{AV} = \sum \underline{T}_{RCS} + \underline{T}_{AERO} \tag{2}$$

$$\underline{A}_C = \underline{F}_{AV}/M_{AV} \tag{3}$$

where

F = Forces

T = Torque

M = Mass

AV = Active vehicle

C = Contact

A = Acceleration

The torque due to the rotational rate of the vehicle, $\underline{T}_{\omega V}$, is calculated and is combined with the torques due to the contact forces and with the torques produced by the gravity gradient, $\underline{T}_{GG}$, calculate the vehicle angular acceleration.

$$\underline{T}_{\omega V} = \underline{\omega}_V \times (\overset{*}{I}\, \underline{\omega}_V) \tag{4}$$

$$\dot{\underline{\omega}}_V = \overset{*}{I}^{-1}(\underline{T}_{AV} - \underline{T}_{\omega V} + \underline{T}_{GG}) \tag{5}$$

where

ω_V is the vehicle angular velocity

$\overset{*}{I}$ is the active I vehicle inertia tensor

The total acceleration ($\underline{A}_T$) of the vehicle due to the contact forces and torques and the vehicle inertia is calculated.

$$\underline{A}_T = \underline{A}_C + \underline{R}_{VX} \times \dot{\underline{\omega}}_V + \underline{\omega}_V \times (\underline{R}_{VX} \times \underline{\omega}_V) \tag{6}$$

The total acceleration of the body is transformed from the vehicle body frame to the inertial frame and then summed with the gravitational accelerations, $\underline{A}_G$,

$$\ddot{\underline{R}}_{IV} = {}^{I}M_V \, \underline{A}_T + \underline{A}_G$$

where Iv denotes inertial vehicle. The matrix ${}^{I}M_V$ transforms from vehicle to inertial reference.

A fourth-order Runge Kutta integrator (DIFEQS for single precision or DPDFQS for double precision) is used to integrate $\ddot{\underline{R}}$ to obtain $\dot{\underline{R}}$ and $\underline{R}$ in ECI coordinates. The time step used is either the active vehicle time step AVSTEP (from ENVCOM), the difference in time required to bring the active vehicle state up to the time required by the rest of the simulation, or TSTEP (passed in ENVCOM), whichever is smaller.

The updated vehicle body rates, $\underline{\omega}_V$, are computed by integrating $\dot{\underline{\omega}}_V$ using a fourth-order Runge-Kutta integrator (DIFEQ for single precision or DPDFQ for double precision). These updated body rates are than used to find the derivative (DAMVI) of the vehicle-to-inertial attitude matrix (AMVI) via

$$\text{DAMVI} = \begin{bmatrix} 0 & -\omega_{V,3} & \omega_{V,2} \\ \omega_{V,3} & 0 & -\omega_{V,1} \\ -\omega_{V,2} & \omega_{V,1} & 0 \end{bmatrix} \text{AMVI} \tag{7}$$

The updated vehicle-to-inertial-attitude matrix is computed by again integrating, using DIFEQ or DPDFQ.

Once the updated attitude matrix has been computed, the Euler angles are found using the Euler angle derivatives (obtained by a call to EULERD or DEULRD) to compute a change in Euler angles (by a call to EULERE or DEULRE). The values of the Euler angles are then updated by adding the changes computed. This procedure requires a call to EXTRCT or DXTRCT.

The integration is repeated until the time associated with the vehicle state equals the time requested.

The active vehicle models have two initialization passes during which the inertial-to-vehicle attitude is initialized, its transpose is computed, and initial values of the environmental forces and torques are computed.

The active vehicle models have no calling arguments and accept no commands.

ACTV1C COMMON BLOCK-VERSION FOR MTV/MEC LENGTH = 150 PAGE 1 OF 1

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE
1	RIV	25	M	INERTIAL TO VEHICLE POSITION VECTOR (4N+1 ARRAY FOR DIFEQ CALLS)
4	DRIV	3	M/S	INERTIAL TO VEHICLE LINEAR VELOCITY VECTOR
10	DDRIV	3	M/S2	INERTIAL TO VEHICLE LINEAR ACCELERATION VECTOR
26	WV	13	RAD/S	ACTIVE VEHICLE ANGULAR VELOCITY VECTOR (4N+1 ARRAY FOR DIFEQ CALLS)
25	DWV	3	RAD/S2	ACTIVE VEHICLE ANGULAR ACCELERATION VECTOR
37	AMVI	37		VEHICLE TO INERTIAL ATTITUDE MATRIX (4N+1 ARRAY FOR DIFEQ CALLS)
[illegible]	DAMVI	9		DERIVATIVE OF VEHICLE TO INERTIAL ATTITUDE MATRIX
74	AVFOR	3	NT	ACTIVE VEHICLE FORCE
77	AVTOR	3	NT*M	ACTIVE VEHICLE TORQUE
82	WVTOR	3	NT*M	TORQUE DUE TO AVTIVE VEHICLE ANGULAR VELOCITY
[illegible]	AVCA	3	M/S2	ACTIVE VEHICLE CONTACT ACCELERATION
[illegible]	AVTA	[illegible]	M/S2	TOTAL ACTIVE VEHICLE LINEAR ACCELERATIONS
[illegible]	GI	3	M/S2	INERTIAL TO VEHICLE GRAVITATIONAL ACCELERATION VECTOR
94	WVI	3	RAD/S	ACTIVE VEHICLE ANGULAR VELOCITY VECTOR IN THE INERTIAL FRAME
97	DWVI	3	RAD/S2	ACTIVE VEHICLE ANGULAR ACCELERATION VECTOR IN THE INERTIAL FRAME
100	AMIV	9		INERTIAL TO VEHICLE ATTITUDE MATRIX VECTOR
109	AIV	3	RAD	ATTITUDE OF VEHICLE IN INERTIAL FRAME
112	DAIV	3	RAD/S	INERTIAL TO VEHICLE EULER ANGLE RATES
115	DDAIV	3	RAD/S2	INERTIAL TO VEHICLE EULER ANGLE ACCELERATION
118	DEIV	3	RAD	INERTIAL TO VEHICLE DELTA EULER ANGLE PER DT
[illegible]	[illegible]	1	S	ACTIVE VEHICLE DELTA TIME STEP
122	AVDTP	1	S	ACTIVE VEHICLE PRINT DT
123	AVTPRT	1	S	ACTIVE VEHICLE PRINT TIME
124	AVPRTC	1		ACTIVE VEHICLE PRINT CONTROL
125	IMRAV	1		RUNGE-KUTTA INDEX FOR ROTATIONAL INTEGRATIONS
126	IMRAV	1		RUNGE-KUTTA INDEX FOR ATTITUDE MATRIX INTEGRATIONS
127	DTSTEP	1	S	DELTA TIME STEP DEFINED BY TSTEP AND AVTIME
1	NONE	3		UNUSED LOCATIONS
131	PAVRDC	1		IF >0, CAUSES SINGLE PASS ACTVEH PRINT
132	PDRIV	3	M/S	PREVIOUS PASS VALUE FOR DRIV
135	PAIV	3	RAD	PREVIOUS PASS VALUE FOR AIV
138	AVTIME	1	S	ACTIVE VEHICLE TIME

4.2 AERO

AERO/AERO50

AERO is the driver routine which calls models of the aerodynamic force and torque for each vehicle in the simulation sequentially. It has no calling arguments and no COMMON block requirements. It is specified on the environment input cards (see section 2).

AERO/AERO37

PROGRAM DESCRIPTION

AERO37 models the aerodynamic forces and torques on an on-orbit vehicle. The vehicle is approximated by three flat plates. In the case of the Space Shuttle Orbiter (SSO), the aerodynamic characteristics modeled are for the 147/140B approximation to the 140 A/B Space Shuttle Vehicle Orbiter (ref. 8). This model includes a calculation to approximate the atmospheric density for the 1962 U.S. Standard Atmosphere (ref. 9).

MATH MODEL

AERO37 approximates an on-orbit aero model for the orbiter using the Newtonian Impact Theory for a flat plate. For the orbit altitude range being considered, the drag coefficient, C_D, is assumed constant. Further, at this altitude, the lift coefficient, C_L, is zero. Therefore, torque on a vehicle can be approximated by modeling the vehicle as a combination of three flat plates and determined from the drag forces acting on each plate (ref. 10).

Calculation of the relative atmospheric density is dependent upon the geometric altitude above mean sea level. For geometric altitudes greater than 500,000 feet, the equation is

$$\rho_f = \rho_o * \exp \left\{ - [24.2 + 0.002889Z - 2605./Z] \right\}$$

or for geometric altitudes from 280,000 to 500,000 feet,

$$\rho_f = \rho_o * \exp \left\{ - [39.57 - 0.01044Z - 6956./Z] \right\}$$

where

ρ_f is the relative atmospheric density in pounds per cubic foot

ρ_o is the atmospheric density at the earth's surface in pounds per cubic foot

Z is the geometric altitude in kilofeet

Then to convert the density to metric units, $\rho = 16.01843\ \rho_f$ kilograms per cubic meter.

The dynamic pressure, q, is defined by $q = 0.5\ \rho V_\infty^2$ newtons per square meter where V_∞ is the magnitude of the inertial velocity in meters per second.

The aerodynamic forces, $\bar{D}_i$, are calculated by

$$\bar{D}_1 = C_D\ q\ A_1 |\hat{k} \cdot \bar{I}_o| \bar{I}_o \text{ newtons}$$

$$\bar{D}_2 = C_D\ q\ A_2 |\hat{j} \cdot \bar{I}_o| \bar{I}_o \text{ newtons}$$

$$\bar{D}_3 = C_D\ q\ A_3 |\hat{i} \cdot \bar{I}_o| \bar{I}_o \text{ newtons}$$

where

$\bar{A}_i$ is a vector normal to the true planform area having a magnitude (A_i) equal to the true planform area

C_D is the drag coefficient

$\bar{I}_o$ is the unit vector of the inertial velocity vector

The vehicle axes with the origin at the vehicle center of mass are defined as

X_V - Out the nose

Y_V - Out the right wing

Z_V - Down

The vector $\bar{A}_1$ is normal to an X-Y plane plate and in the direction of Z. The $\bar{A}_2$ is normal to the X-Z plane plate and in the direction of Y. The $\bar{A}_3$ is normal to a Y-Z plane plate and in the direction of X.

The origins are the centers of pressure of A_1, A_2, and A_3, respectively.

The torques, $\bar{M}_i$, are calculated

$$\bar{M}_1 = \bar{r}_1 \times \bar{D}_1 \text{ newtons-meters}$$

$$\bar{M}_2 = \bar{r}_2 \times \bar{D}_2 \text{ newtons-meters}$$

$$\bar{M}_3 = \bar{r}_3 \times \bar{D}_3 \text{ newtons-meters}$$

Where $\bar{r}_i$ is a position vector from the center of mass to the center of pressure of A_i in meters.

The total aero forces, $\bar{D}_T$, and torques, $\bar{M}_T$, (ignoring shading of one part of the vehicle by another) is

$$\bar{D}_T = \Sigma \bar{D}_i = \Sigma C_D \, q \, |(\bar{A}_i \cdot \bar{I}_o)| \bar{I}_o \text{ newtons}$$

$$\bar{M}_T = \Sigma \bar{r}_i \times \bar{D}_i \text{ newton-meters}$$

INPUT/OUTPUT

This model has no calling arguments.

The following data input is required:

Locations of the center of pressure in the Fabrication frame. ($\bar{R}_{FCP_i}$)

Drag coefficient (C_D)

Magnitudes of the true planform areas (A_1, A_2, A_3)

All the above data can be changed using the SOAP data input scheme.

Input from the active vehicle and mass properties common blocks is required. AERO37 requires on each update from the active vehicle the following:

Inertial position vector (R_{IV})

Inertial velocity vector ($\dot{\bar{R}}_{IV}$)

Vehicle to inertial attitude transformation matrix ($^{I}M_{V}$)

Only on initialization of AERO37 is information required from the mass properties common block. That is

Fabrication to vehicle position vector ($\bar{R}_{FV}$)

Fabrication to vehicle attitude transformation matrix ($^{V}M_{F}$)

The following list gives the program mnemonic, location, and stored value of these variables.

Variable	Symbol	Location	Description	Value	Units
$\bar{R}_{FPC_1}$	AERSTA(1)	7	Station locations of the center of pressure of A1	1111.5	inches
	AERSTA(2)	8		0.0	
	AERSTA(3)	9		363.0	
$\bar{R}_{FPC_2}$	AERSTA(4)	10	Station locations of the center of pressure of A2	1064.4	inches
	AERSTA(5)	11		0.0	
	AERSTA(6)	12		414.4	
$\bar{R}_{FPC_3}$	AERSTA(7)	13	Station locations of the center of pressure of A3	1300.0	inches
	AERSTA(8)	14		0.0	
	AERSTA(9)	15		406.3	
CD	CD	16	Coefficient of drag	2.0	
A1	A1	17	Magnitudes of the true planform areas	397.76	meters^2
A2	A2	18		280.95	
A3	A3	19		99.81	

C-3

This information is used to compute $\bar{r}_1$, $\bar{r}_2$, and $\bar{r}_3$ positions of the center of pressure.

$$\bar{r}_1 = {}^{V}M_F \, (\bar{R}_{FCP_1} - \bar{R}_{FV}) \text{ meters}$$

$$\bar{r}_2 = {}^{V}M_F \, (\bar{R}_{FCP_2} - \bar{R}_{FV}) \text{ meters}$$

$$\bar{r}_3 = {}^{V}M_F \, (\bar{R}_{FCP_3} - \bar{R}_{FV}) \text{ meters}$$

Output from AERO37 is

Sum of the aero forces ($\bar{D}_T$) in newtons

Sum of the aero torques ($\bar{M}_T$) in newtons-meters

Output of these forces and torques is optional and is controlled by an aero print control switch, AEROPC. When AEROPC is 0.0, AERO37 will not print. When AEROPC is 1.0, the output parameters are printed at the specified delta time contained in AERPDT. When AEROPC is 5.0, the output parameters are output every time AERO37 is updated. Starting the print at some time later in the run can be accomplished by setting the desired start time in AEROPT. All print is nominally turned off.

STORED VALUES

AERO37 defines initial values for several parameters located in AEROC common, which may be altered via input cards.

SOURCE

AERO37 was developed from AERO18. Improvements in AERO37 include calculations for three plates instead of two as described in reference 9, and calculation of the atmospheric density as defined in reference 8.

VERSION AVAILABILITY

Both single and double-precision versions of AERO37 are available. Block Data routines (BDAERO) are also available to support them.

PROGRAM VERIFICATION

The AERO37 model subroutine was verified in closed-loop SSFS simulation runs.

AER1

PROGRAM DESCRIPTION

AER1 computes forces and torques on a vehicle on orbit, using the same math model employed by AERO37; the only differences are in the initialization pass and the input data. AER1 bypasses use of a fabrication frame, therefore plate positions, normal vectors, and areas are input directly in vehicle body coordinates. All input and output is identical with that for AERO37 with the exception that $\overline{R}_{FV}$ and $\overline{V}_{MF}$ are not used.

VERSIONS

Both single and double-precision versions are available.

AER1C COMMON BLOCK-VERSION FOR MTV/MEC LENGTH = 63 PAGE 1 OF 1

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE
1	SAEROF	3	NT	AERODYNAMIC FORCE ACTING UPON THE VEHICLE
4	SAEROT	3	NT-M	AERODYNAMIC TORQUES ACTING UPON THE VEHICLE
[illegible]		9		UNUSED
17	AERCCP	9	M	AERODYNAMIC CENTER OF PRESSURE FOR EACH SURFACE
26	FORC1	3	NT	AERODYNAMIC FORCE ACTING UPON SURFACE 1 OF THE VEHICLE
29	FORC2	3	NT	AERODYNAMIC FORCE ACTING UPON SURFACE 2 OF THE VEHICLE
32	FORC3	3	NT	AERODYNAMIC FORCE ACTING UPON SURFACE 3 OF THE VEHICLE
35	H	1	M	GEOMETRIC ALTITUDE ABOVE LOCAL MEAN SEA LEVEL
36	RHO	1	KG/M2	ATMOSPHERIC DENSITY
37	VNU	3		UNIT OF NEGATIVE VEHICLE VEOCITY VECTOR
40	SQRTVS	1	M/S	MAGNITUDE OF VELOCITY VECTOR
41	VSQ	1	M2/S2	VELOCITY VECTOR SQUARED
42	Q	1	KG/M-S2	DYNAMIC PRESSURE
43	DRIVV	3	M/S	VELOCITY VECTOR
46	AERCPC	1		PRINT CONTROL: 0.0 = OFF 1.0 = PRINT EVERY AERPDT 5.0 = PRINT EVERY PASS
47	AERPDT	1	S	AERO PRINT DELTA TIME
48	AERSPT	1	S	AERO PRINT START TIME
49	A1	1	M2	AREA OF SURFACE 1
50	A2	1	M2	AREA OF SURFACE 2
51	A3	1	M2	AREA OF SURFACE 3
52	TORQ1	3	NT-M	TORQUE AT C.P. LOCATION 1
55	TORQ2	3	NT-M	TORQUE AT C.P. LOCATION 2
58	TORQ3	3	NT-M	TORQUE AT C.P. LOCATION 3
61	SF1	1		SCALE FACTOR 1
62	SF2	1		SCALE FACTOR 2
63	SF3	1		SCALE FACTOR 3

4.3 GRAVITY

GRAV2

PROGRAM DESCRIPTION

GRAV2 provides a gravity model for earth orbital operations. Various degrees of fidelity in gravity calculation can be selected. For the lowest degree of fidelity, GRAV2 returns the spherical earth gravity; whereas for the highest, GRAV2 adds perturbations caused by the sun, the moon, and the nonspherical nature of the earth. The user can select any combination of the perturbations caused by the sun, the moon, and the nonspherical nature of the earth to be added to the spherical earth gravity (for example: nonspherical only).

MATH MODEL

Spherical gravity is calculated using

$$\bar{G}_{sp} = \frac{-\mu_e \bar{R}}{|\bar{R}|^3}$$

where

$\bar{G}_{sp}$ is the spherical earth gravitational acceleration.

μ_e is the gravitational constant of the earth.

$\bar{R}$ is the position vector of the point at which the gravitational acceleration is desired, in earth-centered-inertial (ECI) coordinates.

The nonspherical earth perturbation includes up to the fourth harmonic of the earth. The equation used is:

$$\bar{G}_{N-S} = \frac{\mu_e}{|\bar{R}|^5}\left\{\left[FJ(5x-1) + \frac{HS \cdot ZS(7x-3) + FK(6x-a-9x^2)}{|\bar{R}|^2}\right]\bar{R} + \left[-2FJ \cdot ZS + HS(0.6-3x) + \frac{FK \cdot ZS(4x-b)}{|\bar{R}|^2}\right]\bar{Z}\right\}$$

where

$\bar{G}_{N-S}$	is the nonspherical gravity perturbation
μ_e	is the gravitational constant of the earth
$\bar{R}$	is the vehicle position vector in earth-centered inertial (ECI) coordinates
FJ	is a constant relating to the nonsphericity of the earth
HS	is a constant relating to the nonsphericity of the earth
FK	is a constant relating to the nonsphericity of the earth
ZS	is the z-component of the position vector $\bar{R}$
a	is a constant equal to 0.4285714286
b	is a constant equal to 1.714285714
x	is $ZS^2/|\bar{R}|^2$
$\bar{Z}$	is a unit vector in the Z (R(3)) direction

The sun perturbation acceleration is calculated using

$$\bar{G}_s = \frac{\mu_s}{|\bar{r}_{vs}|^3} (q\bar{r}_{es} - \bar{r}_{ev})$$

where

$\bar{G}_s$ is the sun's gravitational perturbation acceleration

μ_s is the gravitational constant of the sun

$\bar{r}_{vs}$ is the vector from the vehicle to the sun

$\bar{r}_{es}$ is the vector from the earth to the sun

$\bar{r}_{ev}$ is the vector from the earth to the vehicle.

and q is defined:

$$q = X \frac{[3 + X(3 + X)]}{[1 + X][(1 + X)^2 + \sqrt{1 + X}]}$$

where

$$X = \frac{2\bar{r}_{vs} \cdot \bar{r}_{ev} + |\bar{r}_{ev}|^2}{|\bar{r}_{vs}|^2}$$

The moon perturbation is the same as above with s subscripts changed to m and the position vectors reflecting the moon's position instead of the sun's.

The units of the symbols in the preceding equations are shown below.

$\bar{R}$	m	a	-
$\bar{G}_{sp}$	m/sec^2	b	-
$\bar{G}_{N-S}$	m/sec^2	x	-
$\bar{G}_s$	m/sec^2	$\bar{Z}$	-
$\bar{G}_m$	m/sec^2	$\bar{r}_{vs}$	m
μ_e	m^3/sec^2	$\bar{r}_{es}$	m
μ_s	m^3/sec^2	$\bar{r}_{ev}$	m
μ_m	m^3/sec^2	$\bar{r}_{vm}$	m
FJ	m^2	$\bar{r}_{em}$	m
HS	m^3	q	-
FK	m^4	X	-
ZS	m		

A complete discussion of the mathematical model appears in reference 11.

All measurements are made in the earth-centered-inertial (ECI) coordinate system.

INPUT/OUTPUT

GRAV2 is called with three calling arguments:

GTIME, $\bar{P}OS$, $\bar{G}RVTY$

where

GTIME is the current time,

$\bar{P}OS$ is the position at which the gravitational acceleration is to be calculated,

$\bar{G}RVTY$ is the gravitational acceleration vector.

GTIME and $\overline{POS}$ are input variables and $\overline{GRVTY}$ is output.

JPERT, a flag located in cell number one of GRAVC COMMON, is used to determine which perturbations (if any) are to be added to the spherical earth gravity. JPERT is an octal integer and can take on any value from 0 to 7. Each of the three least significant bits of JPERT is a flag indicating whether a particular perturbation is to be included in the gravity computations as follows:

1st (least significant) bit	Nonspherical perturbation flag
2nd bit	Sun perturbation flag
3rd bit	Moon perturbation flag

The quantities reflected in the calculation of $\overline{GRVTY}$, depending upon the value of JPERT, are shown below:

JPERT	3rd Bit (MOON)	2nd Bit (SUN)	1st Bit (NONSPH)	$\overline{GRVTY}$ Reflects
0	0	0	0	SPHER only
1	0	0	1	SPHER + NONSPH
2	0	1	0	SPHER + SUN
3	0	1	1	SPHER + NONSPH + SUN
4	1	0	0	SPHER + MOON
5	1	0	1	SPHER + NONSPH + MOON
6	1	1	0	SPHER + SUN + MOON
7	1	1	1	SPHER + NONSPH + SUN + MOON

where SPHER stands for "spherical" and NONSPH stands for "nonspherical."

GRAV2 initially defines the value of JPERT equal to zero. This value may be changed by input if desired. On the initialization pass GRAV2 extracts the perturbation flags from JPERT and stores them in GRAVC COMMON as follows:

GRAVC(2) Nonspherical perturbation flag
GRAVC(3) Sun perturbation flag
GRAVC(4) Moon perturbation flag

GRAV2 does not alter the contents of cells GRAVC(2)-(4) after initialization. Thus, if for any reason, the user desires to alter JPERT after initialization, he must also alter locations two through four of GRAVC to reflect the new value of JPERT.

GRAV2 defines initial values (via DATA statements) for the following common block variables:

JPERT	FJ
μ_e	HS
μ_s	FK
μ_m	

The initial values will appear in the COMMON Block Map section of the SSFS User's Guide. They may be changed by input if desired.

GRAV2 calls one additional subroutine, POSSUM, to calculate the position of the sun and moon when perturbations due to those bodies are requested. Several versions of POSSUM exist. All of them use tabulated data. Typically, they use positions 55 and following in GRAV COMMON.

STORED VALUES

GRAV2 defines initial values for two local variables, a and b, used in the calculation of the gravity perturbation due to the nonspherical nature of the earth.

$$a = 0.4285714286$$

$$b = 1.714285714$$

COORDINATE SYSTEMS

GRAV2 uses the earth-centered-inertial (ECI) coordinate system.

VERSIONS AVAILABLE

Single and double-precision versions of GRAV2 are available.

PROGRAM VERIFICATION

Open-loop checkout of GRAV2 consisted of comparison runs between GRAV2 and the gravity portion of the UNIVRS subroutine used on the AGC bit-by-bit simulator. Test cases were run using the following inputs:

Test Case	PERT (for UNIVRS)	JPERT (for GRAV2)	Time (Sec)	Position Vector (meters)
1	0.	0	50.	6.4E+06, 7.5E+06, -7.0E+06
2	1.	7	50.	6.4E+06, 7.5E+06, -7.0E+06
3	0.	0	150.	-7.0E+06, 5.0E+07, 6.0E+06
4	1.	7	150.	-7.0E+06, 5.0E+07, 6.0E+06
5	0.	0	1000.	8.0E+06, 0.0, 0.0
6	1.	7	1000.	8.0E+06, 0.0, 0.0

The results of the comparison are listed below.

Test Case	Subroutine	Gravitational Acceleration (meters/sec^2)		
1	GRAV2	-1.4429556E+00,	-1.6909636E+00,	1.5782327E+00
	UNIVRS	-1.4429556E+00,	-1.6909636E+00,	1.5782326E+00
2	GRAV2	-1.4425145E+00,	-1.6904477E+00,	1.5791791E+00
	UNIVRS	-1.4425145E+00,	-1.6904477E+00,	1.5791791E+00
3	GRAV2	2.1229797E-02,	-1.5164141E-01,	-1.8196969E-02
	UNIVRS	2.1229796E-02,	-1.5164140E-01,	-1.8196968E-02
4	GRAV2	2.1231229E-02,	-1.5164534E-01,	-1.8196672E-02
	UNIVRS	2.1231228E-02,	-1.5164533E-01,	-1.8196671E-02
5	GRAV2	-6.2281439E+00,	0.0,	0.0
	UNIVRS	-6.2281438E+00,	0.0,	0.0
6	GRAV2	-6.2345735E+00,	1.2922639E-07,	-1.0695097E-05
	UNIVRS	-6.2345736E+00,	1.2922641E-07,	-1.0695097E-05

The above comparison verifies the correctness of the GRAV2 gravity model.

SOURCE

The GRAV2 math model has been extracted from the UNIVRS subroutine of the AGC bit-by-bit simulator.

GRAVC COMMON BLOCK-VERSION GRAV2 LENGTH = 54 PAGE 1 OF 2

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
1	JPERT	1		PERTURBATION FLAG (INIT VALUE= 0) =0 GRAV2 RETURNS SPHERICAL EARTH GRAVITY =1 GRAV2 ADDS PERTURBATION CAUSED BY NON-SPHERICAL NATURE OF THE EARTH =2 GRAV2 ADDS PERTURBATION CAUSED BY SUN =3 GRAV2 ADDS PERTURBATIONS CAUSED BY SUN AND NON-SPHERICAL NATURE OF THE EARTH =4 GRAV2 ADDS PERTURBATION CAUSED BY MOON =5 GRAV2 ADDS PERTURBATIONS CAUSED BY MOON AND NON-SPHERICAL NATURE OF THE EARTH =6 GRAV2 ADDS PERTURBATIONS CAUSED BY SUN AND MOON =7 GRAV2 ADDS PERTURBATIONS CAUSED BY SUN, MOON, AND NON-SPHERICAL NATURE OF EARTH	
2	NONSPH	1		NON-SPHERICAL PERTURBATION FLAG, FIRST (LEAST SIGNIFICANT) BIT OF JPERT =0 NON-SPHERICAL PERTURBATIONS TO BE OMITTED =1 NON-SPHERICAL PERTURBATIONS TO BE INCL	
3	JSUN	1		SUN PERTURBATION FLAG, SECOND BIT OF JPERT =0 SUN PERTURBATIONS TO BE OMITTED =1 SUN PERTURBATIONS TO BE INCLUDED	
4	JMOON	1		MOON PERTURBATION FLAG, THIRD BIT OF JPERT =0 MOON PERTURBATIONS TO BE OMITTED =1 MOON PERTURBATIONS TO BE INCLUDED	
5	FMU	3	M**3/S**2	GRAVITATIONAL CONSTANTS OF EARTH, SUN, MOON (INIT VALUE= 3.98601199995E+14, 1.32712498999E+20, 4.90277999997E+12)	
8	FJ	1	M**2	CONSTANT RELATING TO NON-SPHERICITY OF EARTH (INIT VALUE= 6.6[illegible]434476667E+10)	
9	HS	1	M**3	CONSTANT RELATING TO NON-SPHERICITY OF EARTH (INIT VALUE= -1.49194932892E+15)	
10	FK	1	M**4	CONSTANT RELATING TO NON-SPHERICITY OF EARTH (INIT VALUE= 1.3932634[illegible]543E+22)	
11	TSAV	1	S	TIME OF LAST CALL TO SUBROUTINE POSSUM	
12	RSCOM	3	M	POSITION VECTOR FROM EARTH TO SUN	I
15	RMCOM	3	M	POSITION VECTOR FROM EARTH TO MOON	I
18	FM	1	RAD	MEAN ANOMALY OF EARTH ORBIT	
19	FMDR	1	RAD/S	MEAN ANGULAR MOTION OF THE EARTH (INIT VALUE= 1.99[illegible]98660882E-07)	
20	TAU	1	S	TIME OF PERIHELION PASSAGE FOR THE EARTH, MEASURED FROM JULY 1, 1971 (INIT VALUE= 1.61343944093E+07)	
21	EC	1		ECCENTRICITY OF EARTH ORBIT (INIT VALUE= 1.67238789670E-02)	
22	ES	1	RAD	ECCENTRIC ANOMALY OF EARTH ORBIT	

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE	C
23	DW	1	RAD	ITERATIVE CHANGE IN ECCENTRIC ANOMALY OF EARTH ORBIT	
24	AS	1	M	MEAN DISTANCE FROM EARTH TO SUN (INIT VALUE= 1.49597892998E+11)	
25	ONESP	3		UNIT VECTOR FROM SUN TO PERIHELION OF EARTH ORBIT (INIT VALUE= -2.15738559014E-01, 9.76451163203E-01, 0.0E+00)	
28	ONESN	3		UNIT VECTOR FROM SUN TO ASCENDING NODE OF EARTH ORBIT (INIT VALUE= -9.76451163203E-01, -2.15738559014E-01, 0.0E+00)	I
31	FOBLZ	1	RAD	OBLIQUITY OF THE ECLIPTIC AT START OF BESSELIAN YEAR, JAN. 1, 1972 (INIT VALUE= 4.09156217531E-01)	
32	FOBLR	1	RAD/S	TIME RATE OF CHANGE OF OBLIQUITY OF THE ECLIPTIC AT START OF BESSELIAN YEAR, JAN.1,1972 (INIT VALUE= -7.19758579146E-14)	
33	FOBO	1	RAD	OBLIQUITY OF ECLIPTIC AT CURRENT TIME	
34	DJD	1	S	TIME FROM JULY 1, 1971 TO JAN. 1, 1972 (INIT VALUE= 1.59193404001E+07)	
35	SF	1		MOON/EARTH MASS RATIO (INIT VALUE= 1.21505206497E-02)	
36	SQEC	1		SQUARE ROOT OF 1-EC**2	
37	RO	3	M	POSITION OF MOON AT TIME ZERO, JULY 1, 1971 (INIT VALUE= -3.25575108494E+08, -1.96273145146E+08, -1.23982936816E+08)	I
40	A	3	M/S	VECTOR COEFFICIENT IN INTERPOLATION FORMULA FOR POSITION OF MOON (INIT VALUE= 2.74449360424E+02, -3.67258466756E+02, -1.68424181413E+02)	I
43	B	3	M/S**2	VECTOR COEFFICIENT IN INTERPOLATION FORMULA FOR POSITION OF MOON (INIT VALUE= 2.74301336717E-04, -4.43546113UE-04, 8.51388098524E-05)	I
46	C	3	M/S**3	VECTOR COEFFICIENT IN INTERPOLATION FORMULA FOR POSITION OF MOON (INIT VALUE= -4.94120003374E-11, 1.03441134091E-10, 4.95626137316E-11)	I
49	D	3	M/S**4	VECTOR COEFFICIENT IN INTERPOLATION FORMULA FOR POSITION OF MOON (INIT VALUE= -3.22391499194E-17, -5.84027400497E-16, -5.33626863UUE-18)	I
52	W	3	M/S**5	VECTOR COEFFICIENT IN INTERPOLATION FORMULA FOR POSITION OF MOON (INIT VALUE= -1.2198277E-24, -3.8418349E-24, -2.0733522E-24)	I

For locations 55 and subsequent, see POSSUM.

4.4 GRAVITY GRADIENT

GGT1

PROGRAM DESCRIPTION

GGT1 models the generalized gravity gradient torque, which is produced about the vehicle's center of mass by the earth's gravitational acceleration acting on heterogeneous and asymmetrical vehicles. A spherical earth is used since the torque produced due to the earth's oblateness is negligible.

MATH MODEL

The gravity gradient torque components (T_x, T_y, T_z) about the vehicle reference axes are

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = -\frac{3\mu}{|\bar{R}|^3} \begin{bmatrix} -D_xD_yI_{xz} + D_xD_zI_{xy} - (D_y^2 - D_z^2)I_{yz} + D_yD_z(I_{yy} - I_{zz}) \\ -D_yD_zI_{xy} + D_xD_yI_{yz} - (D_z^2 - D_x^2)I_{xz} + D_xD_z(I_{zz} - I_{xx}) \\ -D_xD_zI_{yz} + D_yD_zI_{xz} - (D_x^2 - D_y^2)I_{xy} + D_xD_y(I_{xx} - I_{yy}) \end{bmatrix}$$

where μ = earth's gravitational constant [PS:EGC]

$\bar{R}$ = vehicle inertial position vector expressed in the vehicle body frame [PS:R]

I_{qq} = elements of vehicle inertia tensor in the vehicle center of gravity originated system. [PS:VI]

D_q = direction cosines between the radius vector R and the vehicle body axes. [PS:D]

The equations above use the negative products of inertia. All calculations are made in the vehicle body frame. In the above [PS:] indicates program symbol.

INPUT/OUTPUT

GGT1 has two calling arguments.

$\bar{R}$ = inertial state vector expressed in vehicle body coordinates. (meters)

I_{qq} = vehicle inertia tensor ($kg-m^2$)

One additional parameter is required for input and is located in locations 7 through 12 of the GGT1.

$\bar{AM}$ = gravity gradient torque axial multiplies (one for each axis), which are used to scale or turn off the torque about a particular axis. (nominally set to 1)

Output from GGT1 is

T = gravity gradient torque [PS:GGT] (M-NT) about the vehicle center of gravity (M-Nt). [PS:GGT]

VERSIONS AVAILABLE

Both single and double-precision versions are available.

PROGRAM VERIFICATION

The verification of GGT1 consisted of running several test cases and making hand calculations. One test case was run for a complete circular orbit to illustrate the values obtained from GGT1. In all cases, the same inertia tensor was used.

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} = \begin{bmatrix} 2725000.0 & -20000.0 & -333000.0 \\ -20000.0 & 19883000.0 & 5000.0 \\ -333000.0 & 5000.0 & 21495000.0 \end{bmatrix} \text{ slugs-ft}^2$$

A spherical gravity was used and the initial inertial state was defined using

$$\underline{R}_{IV} = 6600000.0,\ 0.0,\ 0.0 \text{ meters}$$

$$\underline{V}_{IV} = 0.0,\ 7770.0,\ 0.0 \text{ meters/second}$$

In this case, the GGT1 routine calculated the values in table 4-I for the circular orbit of 5400 seconds. The GG Torque plot samples the Torque vector every 20 seconds.

TABLE 4-I.- GRAVITY GRADIENT TORQUE VECTOR

Run Time (Sec)	GGT_x (NT-M)	GGT_y (NT-M)	GGT_z (NT-M)
0	- .06998861	44.89743710	- .04040712
600	3.11959702	29.31005049	-39.59090662
1200	-1.31008284	-45.24714470	-24.70406365
1800	.17021475	-22.90914392	25.18761992
2400	-1.40059164	5.56837219	47.49012947
3000	1.96025780	-27.64657974	39.47186518
3600	.05934277	- 3.77952236	- 3.82524449
4200	- .84780970	49.77582121	5.13785291
4800	-3.68393409	22.61737800	35.36430979
5400	.43922012	-29.66196251	27.02822924

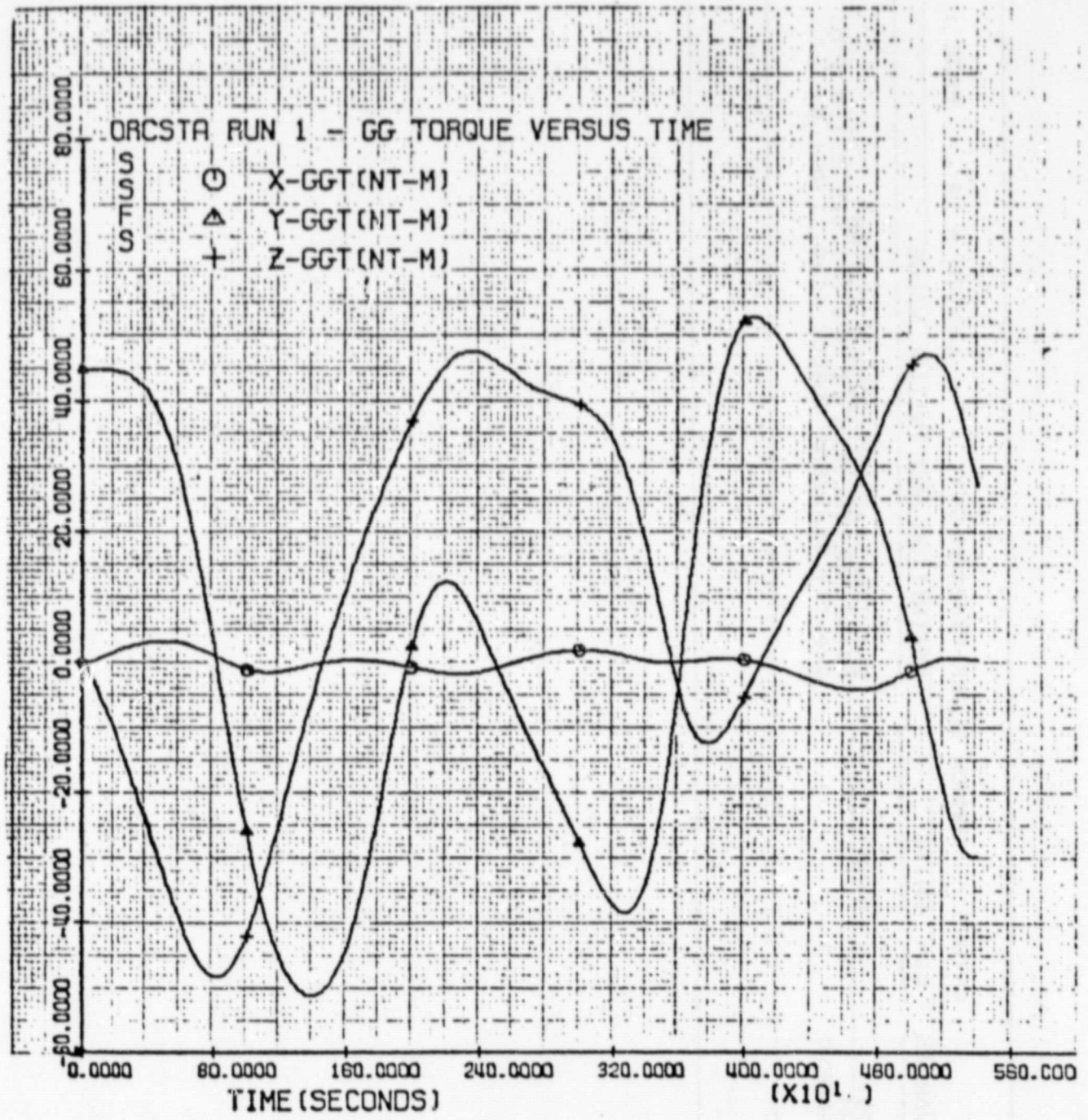


Figure 4-1.- Time history of gradient gravity torque.

GGTORC COMMON BLOCK-VERSION GGT1 LENGTH = 14 PAGE 1 OF 1

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE
1	GGT	6	NT*M	GRAVITY GRADIENT TORQUE VECTOR
	AM	6		AXIAL MULTIPLIERS (1.0, 1.0, 1.0, 1.0, 1.0, 1.0)
13	GGTFLG	2		GRAVITY GRADIENT TORQUE FLAG (1.0, 1.0)

4.5 GYRO

TBD

4.6 HORIZON SENSOR

HORIZ

Subroutine name: HORIZ1, HORIZ2

PROGRAM DESCRIPTION

These routines model the performance of a Barnes Model 13-166-9. This apparatus is pictured in the figure below. It senses pitch and roll at a frequency of 12.5 Hz. If the vehicle is either pitched or rolled more than 5° from local vertical, the sensor output is 5°. No other inaccuracies are modeled. Yaw is not sensed.

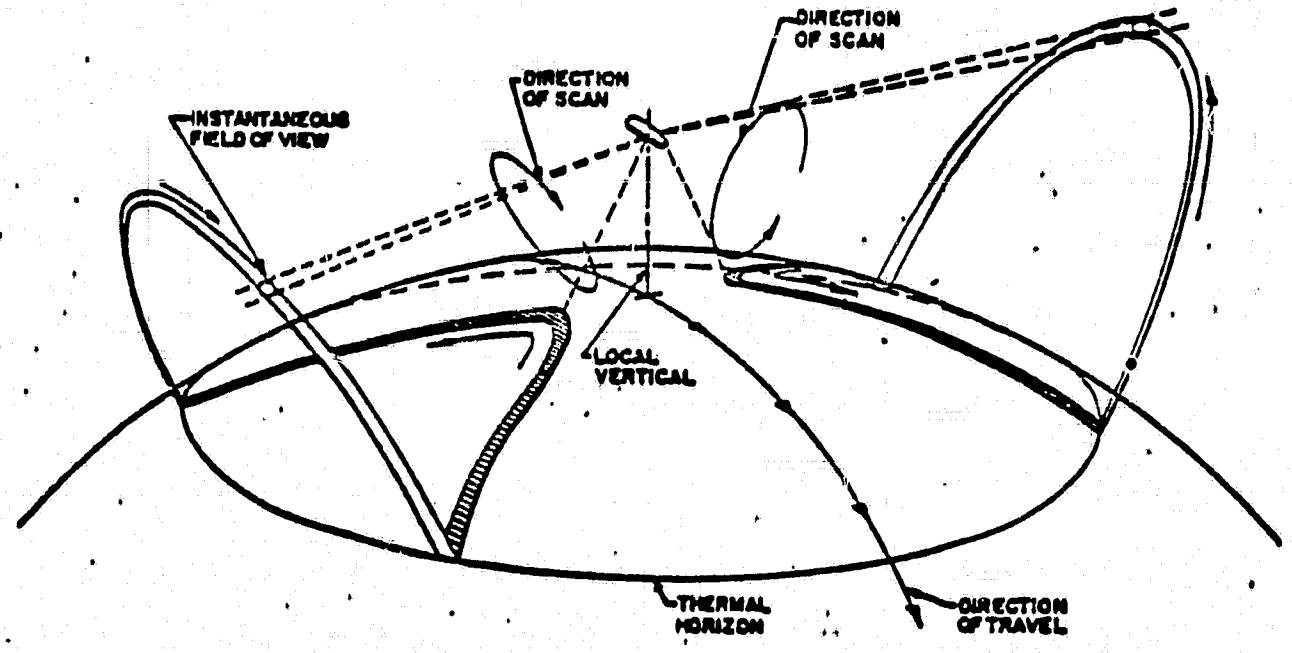


Figure 4-2.- Horizon scan configuration.

MATH MODEL

The horizon sensor model obtains a unit vector antiparallel to the local vertical, ZLVM, by transforming RIV, the position vector of the vehicle in inertial coordinates, to body coordinates via the transformation AMIV

$$ZLVM = AMIV \cdot Unit\ (RIV) \qquad (1)$$

The dot product of the body X vector with the local vertical vector gives the cosine of the angle between the two. If pitch equals zero, this angle should be 90°. For negative pitch, the angle is greater than zero, so the pitch in degrees (PHS) is found to be

$$PHS = 90^{\circ} - \frac{\pi}{180} \cdot \cos^{-1}(ZLVM(1)) \qquad (2)$$

Similarly, for roll (RHS)

$$RHS = 90^{\circ} - \frac{\pi}{180} \cdot \cos^{-1}(ZLM(2)) \qquad (3)$$

If either pitch or roll exceeds 5° in magnitude, it is limited to 5° and the sign is retained.

This model has no calling arguments.

INPUT/OUTPUT

The symbol # denotes a vehicle number.

Prog. symbol	I/O	COMMON block name	COMMON block location	D'men.	Description (units)
RIV	I	ACTV#C	1	3	Inertial to vehicle position vector (m)
AMIV	I	ACTV#C	100	9	Inertial to vehicle attitude transformation matrix
LCBHS#	I*	LECCOM # = 1 # = 2	 1348 1406	1	Horizon sensor COMMON block length
IFGHS#	I*	FS#C	70	1	Horizon sensor on flag 1 = on 2 = off

*Input required to use this model.

Prog. symbol	I/O	COMMON block name	COMMON block location	Dimen.	Description (units)
RHS	O	HS#C	1	1	Roll angle (deg)
PHS	O	HS#C	2	1	Pitch angle (deg)
ZLVM	O	HS#C	3	3	Unit vector from origin of ECI coordinates to vehicle in the vehicle body coordinate system.

MASPRO

MASPRO is the mass properties driver. It calls mass properties routines MASPRØ, MASPR1 ... , etc. It maintains no clock and has no COMMON block. There are no calling arguments.

MASPRØ

PROGRAM DESCRIPTION

MASPRØ defines the mass properties of the Orbiter with an aft payload for Operation Mission 2 (sortie). Values are chosen at pre-OMS deorbit burn as in reference 1. The center of mass and vehicle inertia are held constant while the mass can be decremented to model fuel consumption by the RCS jets and main propulsion engines. The nominal mass properties represent the Orbiter with a 32,000 pound payload. An optional mass properties data set may be selected to represent the Orbiter with no payload.

MATH MODEL

Except for the change in mass, MASPRØ performs all its calculations during the initialization pass only.

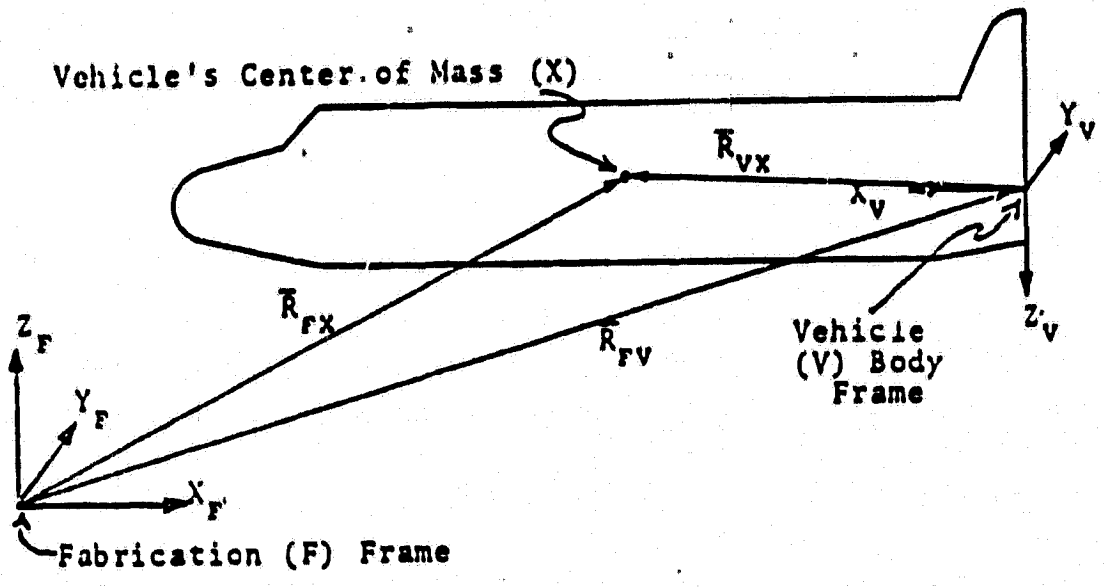
The structural characteristics of the Orbiter vehicle are defined in the fabrication (F) coordinate frame. The location of the vehicle's center of mass (X) frame and the body (V) frame are expressed in the fabrication frame. This mass properties model calculates the position of the vehicle's center of mass in the body frame.

Initial parameters are input to MASPRØ via data statements in English units and converted internally to the MKS system.

The attitude of the vehicle (V) body frame in the fabrication (F) frame,

$$\bar{A}_{FV} = (0.0,\ 180.0,\ 0.0) \text{ degrees} \qquad (1)$$

is converted to radians to define the fabrication to vehicle body frame transformation matrix, $V^*_{M_F}$.



The vector $\bar{R}_{VX}$ is the position vector, expressed in the vehicle system, from the origin of the vehicle coordinate system to the actual center of gravity. The equation used to define $\bar{R}_{VX}$ is

$$\bar{R}_{VX} = \left[V^{*}_{M_F}\right]\left(\bar{R}_{FX} - \bar{R}_{FV}\right) \qquad (2)$$

where

$\bar{R}_{FX}$ is the position of the vehicle's center of mass (X) in the fabrication (F) coordinates.

and

$\bar{R}_{FV}$ is the position of the vehicle's (V) body frame in the fabrication (F) coordinates.

The inertia input is in fabrication coordinates and the product elements are defined by positive integrals. However, the inertia tensor as used by the vehicle routine is referenced to vehicle coordinates and the product elements are defined as negative integrals. Therefore, MASPRØ performs the

following two operations on the input inertia elements to get the output inertia tensor:

1. Negates the product of inertia elements
2. Transforms from fabrication to vehicle coordinates,

$$I_V^* = \left[V_{M_F}^*\right] I_F^* \left[V_{M_F}^*\right]^T \tag{3}$$

The inverse of the inertia tensor is calculated, completing the initial call.

On subsequent calls to MASPRØ, the current Orbiter mass is computed as follows:

$$m = m_0 - \Delta m_{RCS} - \Delta m_{THRUST} \tag{4}$$

where

m	is the current vehicle mass (kg)
m_0	is the initial vehicle mass (kg)
Δm_{RCS}	is the total mass of RCS fuel consumed (kg)
Δm_{THRUST}	is the total mass of main propulsion engine fuel consumed (kg)

INPUT/OUTPUT

This mass properties model requires the following input parameters:

AVMASS - Active vehicle mass; input units are pounds, which are converted to kilograms for simulator calculations and output

RFX - Position of the center of mass (X) in the fabrication frame; input units are inches, which are converted to meters for simulator calculations and output

RFV - Position of the vehicle (V) body frame in the fabrication (F) frame; input units are inches, which are converted to meters for simulator calculations and output

AFV - Attitude of the vehicle (V) body frame in the fabrication (F) frame; input units are degrees which are converted to radians for simulator calculations and output

AVINRT - Active vehicle inertia tensor, input as positive integrals in slugs-ft^2 in the fabrication system. Converted to negative integrals in kg-m^2 in vehicle system for SOAP calculation and output

MPMAS - Mass properties model activity switch (nominally set to -1 for a constant mass model)

NOLOAD - PAYLOAD selection switch, nominally set to 0 for 32,000 pound payload, or input as nonzero for no payload

The above input parameters are initially input in English units and have been defined using data statements. However, the internal system of units for MASPRØ is MKS. Payload independent values are as follows:

$$\begin{aligned} AFV &= 0.0 \quad \text{rad} \quad (0.0^{o}) \\ &\quad 3.1415927 \quad \text{rad} \quad (180.0^{o}) \\ &\quad 0.0 \quad \text{rad} \quad (0.0^{o}) \\ MPMASS &= -1 \end{aligned} \tag{5}$$

Values for both nominal (with payload) and off-nominal (no payload) configurations are given in the attached tables.

If a decremented vehicle mass is desired, i.e., to account for fuel burned by the RCS and main propulsion systems, the model activity switch (ENVCOM location 389) must be changed from -1 to +1. This is done by resetting MPMAS via data input.

The above data values were taken from reference 12. These values can be changed by input if desired. Common block locations may be obtained from the attached listing of the MASPRØ common block.

The output from MASPRØ is:

Parameter	Description	Units
AVMASS	Active vehicle mass	kg
RVX	CM location in vehicle body frame	m
AVINRT	Active vehicle inertia tensor	$kg\text{-}m^2$
AVINV	Inverse of active vehicle inertia tensor	$1/kg\text{-}m^2$
RFV	Position of V in F	m
AMFV	F to V attitude matrix	-
AFV	F to V attitude	rad
RFX	Position of CM in F	m
AVMASI	Initial active vehicle mass	kg

MASPRØ accepts no commands and has no calling arguments.

PROGRAM VERIFICATION

MASPRØ has been verified by open loop checkout runs.

NOMINAL VALUES 32,000 lb PAYLOAD

AVMASS =	91,468.579	kg	(201,653.7 lb)
RFX =	28.5648	m	(1,244.6 in.)
	.0102	m	(0.4 in.)
	9.3269	m	(367.2 in.)
RFV =	28.5648	m	(1,244.6 in.)
	.0102	m	(0.4 in.)
	9.3269	m	(367.2 in.)

AVINRT =

$$\begin{bmatrix} 1{,}134{,}393.750 & -10{,}661.068 & -301{,}528.469 \\ -10{,}661.068 & 8{,}178{,}828.000 & -2{,}450.370 \\ -301{,}528.469 & -2{,}450.370 & 8{,}719{,}432.625 \end{bmatrix}$$

$kg\text{-}m^2$
negative integral
vehicle frame
(output)

$$\begin{bmatrix} 836{,}685.9 & -7{,}863.2 & 222{,}396.0 \\ -7{,}863.2 & 6{,}032{,}394.0 & -1{,}807.3 \\ 222{,}396.0 & -1{,}807.3 & 6{,}431{,}123.5 \end{bmatrix}$$

$slug\text{-}ft^2$
positive integrals
fabrication frame
(input)

NOLOAD = 0

OFF-NOMINAL VALUES NO PAYLOAD

AVMASS =	76,953.623	kg	(169,653.7 lb)
RFX =	28.5877	m	(1,125.5 in.)
	.0127	m	(0.5 in.)
	9.5987	m	(377.9 in.)
RFV =	28.5877	m	(1,125.5 in.)
	.0127	m	(0.5 in.)
	9.5987	m	(377.9 in.)

AVINRT =

$$\begin{bmatrix} 1{,}058{,}958.203 & -9{,}977.057 & -281{,}892.281 \\ -9{,}977.057 & 7{,}992{,}227.437 & -2{,}281.842 \\ -281{,}892.281 & -2{,}281.842 & 8{,}297{,}093.312 \end{bmatrix}$$

$kg\text{-}m^2$ negative integrals vehicle frame (output)

$$\begin{bmatrix} 781{,}047.5 & -7{,}358.7 & 215{,}288.7 \\ -7{,}358.7 & 5{,}894{,}764.5 & -1{,}683.0 \\ 215{,}288.7 & -1{,}683.0 & 6{,}119{,}622.0 \end{bmatrix}$$

$slug\text{-}ft^2$ positive integrals fabrication frame (input)

NOLOAD = 1

MASPRS

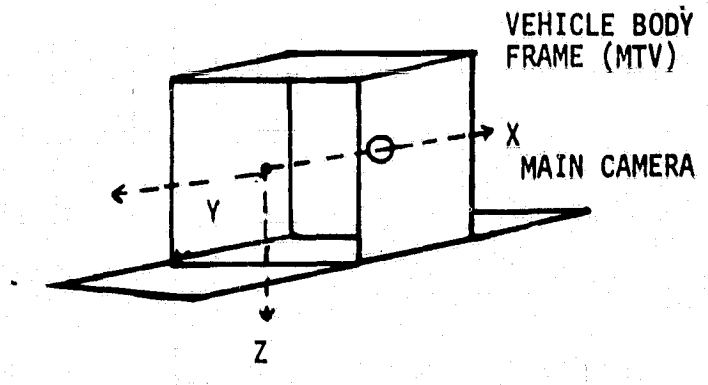
PROGRAM DESCRIPTON

MASPRS defines the mass properties of the maneuverable television (MTV) or any gas-fueled vehicle. The center of mass (CM), the vehicle inertia matrix, and the mass are updated to model fuel consumption by the RCS jets.

MATH MODEL

MASPRS calculates a center of gravity (CG), an active vehicle inertia tensor, and an active vehicle mass each time it is called. These calculations are performed after setting their initial values to zero each time. This mass properties model calculates the vehicle's CG position in the body frame.

Initial parameters are input to MASPRS via data statements in MKS units.



RVX is the position vector of the CG expressed in the vehicle body coordinate system, from the origin of the vehicle coordinate system to the actual CG. The inertia input is in body frame coordinates and the off-diagonal (product) elements are defined by negative integrals. The vehicle is broken into several subvehicles or pods, each with its own inertia tensor. The mass of each pod may change, but the relative magnitudes of the elements of the inertia tensor of each pod remain the same. The initial inertia tensor associated with each pod is also input and defined by the same integrals as in the active vehicle inertia tensor. As the mass of each pod changes so does its inertia tensor, thereby changing the entire active vehicle inertia tensor. Therefore MASPRS performs two operations to obtain the output inertia tensor: (1) Calculates current mass of each pod, and (2) Calculates current inertia tensor of each pod.

A total vehicle inertia tensor is then found and the inverse of the inertia tensor is calculated by a call to INVRSE.

Therefore, on each call to MASPRS the current inertia tensor and its inverse are calculated.

The current vehicle mass is also computed as follows:

$$M = \Sigma M_p \tag{1}$$

where M_p is the mass associated with each pod and

$$M_p = M_{po} - \Delta M_{RCS} \tag{2}$$

where M_{po} is the initial mass associated with each pod and ΔM_{RCS} is the mass of RCS fuel consumed by each pod.

The position of the vehicle CM is recomputed at every pass using

$$\overrightarrow{RVX} = \Sigma M_p \overrightarrow{RVPOD} \tag{3}$$

where the vectors $\overrightarrow{RVPOD}$ are positions of the pod CM in the vehicle body frame. These positions are assumed to be unchanging although the mass associated with each pod can change.

By convention, the pod with index 1 corresponds to those parts of the vehicle which do not move, while pods 2 through 5 represent RCS fuel tanks.

INPUT/OUTPUT

Prog. Symbol	COMMON block name	COMMON block location	Dimen.	Description (unit)
AVMASS	MASP#C	1		Active vehicle mass (kg)
RVX	MASP#C	2	3	Position of CM in vehicle system coordinates (m)
AVINRT	MASP#C	5	9	Active vehicle inertial tensor, input as negative integral in vehicle system coordinates ($kg\text{-}m^2$)
XMASO	MASP#C	23	5	Initial mass of each pod (kg)
RVPOD	MASP#C		3,5	CM of each pod, in vehicle body frame coordinates
FUELUS	RCS#C	8	50	Total fuel used by each jet (kg)
JETPOD	RCS#C	1133	50	An array which identifies pod membership of each RCS jet

INPUT

Prog Symbol	COMMON block name	COMMON block location	Dimen.	Description (unit)
AINRT	MASP#C	43	9,5	Inertia tensor associated with each pod ($kg\text{-}m^2$)

OUTPUT

Prog Symbol	COMMON block name	COMMON block location	Dimen.	Description (unit)
AVINV	MASP1C	14		Inverse of active vehicle inertia tensor ($1/kg\text{-}m^2$)

MASPRS accepts no commands and has no calling arguments.

The input/output for MASPRS/MEC is identical to that for MASPRS with the following exceptions:

INPUT

Prog. symbol	COMMON block name	COMMON block location	Dimen.	Description (units)
*RADGYR	MASP#C	43	9,5	Radius of gyration squared associated with each pod (m^2)
OBJVOL	MASP#C	95	1	Volume of object (MEC without antenna) (m^3)
SPHVOL	MASP#C	96	1	Volume of spherical gas bottles (m^3)

*RADGYR is defined as follows:

$$r_{ij}^2 = \frac{I_{i,j}}{m_{i,j}}$$

where I is the moment of inertia and m is the mass of each pod

MASPRS/MEC

MASPRS/MEC is a mass property version similar in description to MASPRS except for the following changes and assumptions:

a. Two spherical gas bottles, volumes stored in SPNVOL, and treated as POD #2 and POD #3

b. One antenna whose radius of gyration and mass is treated as POD #4, (RADGYR(J,4), XMASS(4))

c. Rest of MEC is an object of volume OBJVOL, which includes the gas bottle volume but not the antenna. It is treated as POD #1, (RADGYR(J,1), XMASS(1)), with the calculation for its radius of gyration using a homogeneous body formula.

VERSIONS AVAILABLE

Both single- and double-precision versions of MASPRS are available. The supporting routine BDMSP#C has both single- and double-precision versions, as well as versions for several vehicles.

MASP#C COMMON BLOCK-VERSION FOR MTV LENGTH = 90 PAGE 1 OF 1

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE
1	AVMASS	1	KG	ACTIVE VEHICLE MASS
2	RVX	3	M	POSITION OF VEHICLE CENTER OF MASS IN THE VEHICLE BODY FRAME
5	AVINRT	9	KG-M2	ACTIVE VEHICLE INERTIA TENSOR
14	AVINV	9	1/KG-M2	INVERSE OF ACTIVE VEHICLE INERTIA TENSOR
23	XMASO	5	KG	INITIAL MASS ASSOCIATED WITH EACH POD
28	RVPOD	3,5	M	CENTER OF MASS OF EACH OF 5 PODS IN VEHICLE BODY FRAME COORDINATES
43	AINRT	9,5	KG-M2	INERTIA MATRIX ASSOCIATED WITH EACH POD

MASP#C COMMON BLOCK-VERSION FOR DEC LENGTH = 100 PAGE 1 OF 1

LOC	VARNAM	DIM	UNITS	DEFINITION AND INITIAL VALUE
1	AVMASS	1	KG	ACTIVE VEHICLE MASS
2	RVX	3	M	POSITION OF VEHICLE CENTER OF MASS IN THE VEHICLE BODY FRAME
5	AVINRT	9	KG-M2	ACTIVE VEHICLE INERTIA TENSOR
14	AVINV	9	1/KG-M2	INVERSE OF ACTIVE VEHICLE INERTIA TENSOR
23	XMASO	5	KG	INITIAL MASS ASSOCIATED WITH EACH POD
28	RVPOD	3,5	M	CENTER OF MASS OF EACH OF 5 PODS IN VEHICLE BODY FRAME COORDINATES
43	RADGYR	9,5	M2	RADIUS OF GYRATION SQUARED OF EACH POD
93	OBJVOL	1	M3	VOLUME OF OBJECT(DEC WITHOUT ANTENNA)
96	SPHVOL	1	M3	VOLUME OF SPHERICAL GAS BOTTLES
97	CUBVOL	1	M3	OBJECT VOLUME MINUS SPHERICAL VOLUME
98	XMASSE	1	KG	OBJECT MASS TIMES OBJECT VOLUME DIVIDED BY CUBE VOLUME
99	ANMASS	1	KG	MASS OF ANTENNA

4.8 REACTION CONTROL SYSTEM

RCS

This routine calls subroutines which model the forces and torques due to each vehicle in the simulation, i.e., RCS0, RCS1, RSC2, ... It has no calling arguments, input or output variables, or COMMON blocks.

RCS0

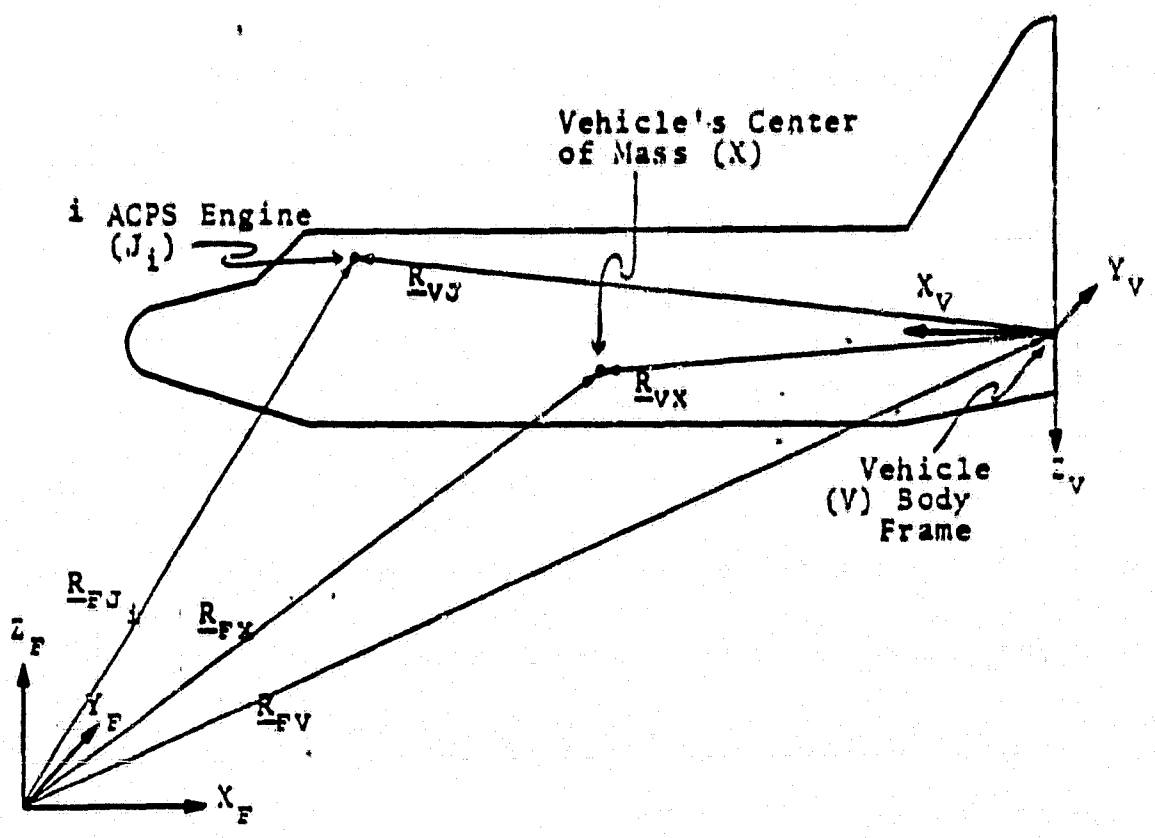
PROGRAM DESCRIPTION

The RCS0 subroutine provides a dynamic model of 38 Attitude Control Propulsion System (ACPS) bipropellant primary thrusters (870-lb nominal vacuum thrust) and six vernier thrusters (25-lb nominal vacuum thrust). These thrusters are arranged as specified in references 13 and 14 for the Rockwell International orbiter vehicle. Each of the 44 thrusters is individually controlled and commanded. Each type of thruster, either primary or vernier, has the same on-delays, off-delays, minimum impulse, specific impulse, total impulse and fuel-loss-per-firing. The forces, torques, and fuel are calculated for each engine. The resultant forces and torques, including vacuum impingement increments, are calculated and filed for incorporation in the vehicle's dynamics. The total fuel consumed is calculated and filed for updating the vehicle's mass properties.

MATH MODEL

RSC0 models 38 ACPS engines of 870-lb thrust each and six vernier engines of 25-lb thrust each, according to the June 1976 data given in reference 15. With appropriate input, RSC0 can model up to 50 ACPS engines which have constant thrust magnitude. Also, up to three different-size engines can be modeled.

The force $(\underline{Q}_i)$ direction (see fig. and table 4-II) depends on the location of the engine on the vehicle. All engine locations are initially defined in the fabrication (F) frame, from which the locations of the ACPS engines in the vehicle's (V) body frame are calculated during initialization. Knowing the vehicle's initial center of mass (X), the torques $(\underline{N}_i)$ of the engines can be calculated.



If the mass properties of the vehicle are constant, then the position ($\underline{R}_{VX}$) of the vehicle's center of mass will be constant, and the torque ($\underline{N}_i$) calculated during initialization will be constant. If the mass properties are not constant, then the ACPS engine torques and torque impingement increments are recalculated each update.

$$\underline{R}_{VJ_i} = {}^{V}M_F\left(\underline{R}_{FJ_i} - \underline{R}_{FV}\right)$$

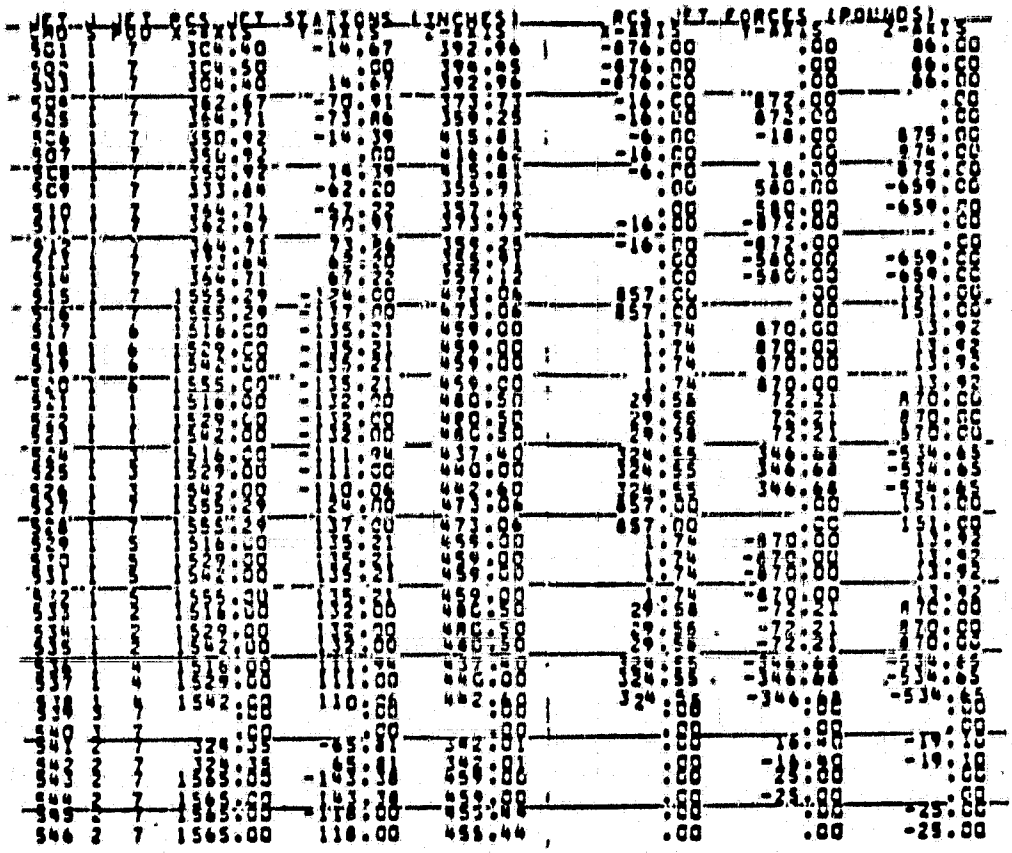
$$\underline{N}_i = \left(\underline{R}_{VJ_i} - \underline{R}_{VX} \times \underline{Q}_i\right)$$

The symbols in the above equations are defined below. The program symbol is given in the brackets [ ].

$\underline{R}_{FV}$ - Position of the V body frame in the F frame [RFV]

$\underline{R}_{VX}$ - Position of X in the V body frame [RVX]

TABLE 4-II.- RCSO INTERNAL VALUES FOR JET SIZES, STATIONS, AND FORCES



Note:
1. RCS Jet Stations $\left(\underline{R}_{FJ_i}\right)$ are given in the fabrication (F) frame, in inches.
2. RCS Jet Forces $\left(\underline{Q}_i\right)$ are given in the vehicle (V) frame. The forces shown do not include impingement increments.
3. RCS Jet Stations for all forward jets (JETPOD = 7) include the distance ℓ [DISTL] from the thruster attach point to the point of effective thrust due to nozzle scarfing effects.

V_{M_F} Transformation matrix from F frame to V frame

$\underline{R}_{FJ_i}$ - Position (station) of the ith ACPS engine (J) in the F frame refiled from RCSSTI [RCSSTA]

$\underline{R}_{VJ_i}$ - Position of J in the V body frame [RCSPOS]

$\underline{Q}_i$ - Force vector of the ith ACPS engine refiled in newtons from RCSFI [RCSFOR]

$\underline{N}_i$ - Torque vector of the ith ACPS engine [RCSTOR]

Note: For the 14 main and two vernier jets in the forward jetpod, a distance ℓ [DISTL(I)] (see table 4-III) is added to $\underline{R}_{FJ_i}$. This is the distance from the jet attach point to the effective thrust point as a result of nozzle scarfing effects.

TABLE 4-III.- DISTL

DISTL contains the distance (in inches) from the thruster attach point (RCSSTI) to the effective thrust point for all forward RCS engines, the last six locations are for the verniers. Jets 509, 510, 513, 514, 541, 542 include cant angles (37 ^{o}).

JET ID	X	Y	Z
501	-22.88	0.0	0.0
502	-22.88	0.0	0.0
503	-22.88	0.0	0.0
504	0.0	-15.28	0.0
505	0.0	-15.28	0.0
506	0.0	0.0	16.22
507	0.0	0.0	15.80
508	0.0	0.0	16.22
509	0.0	-12.385	-16.436
510	0.0	-12.385	-16.436
511	0.0	15.28	0.0
512	0.0	15.28	0.0
513	0.0	12.385	-16.436
514	0.0	12.385	-16.436
541	0.0	-11.976	-15.893
542	0.0	11.976	-15.893

The fuel mass consumed by the firing of an engine is computed using a variable specific impulse. This variable specific impulse is a function of the electrical pulse width (EPW). EPW is defined as

$$EPW = COMMAND_{OFF} - COMMAND_{ON}$$

where

$COMMAND_{ON}$ - Time at which the engine is commanded on (does not include an on-delay)

$COMMAND_{OFF}$ - Time at which the engine is commanded off (does not include an off-delay)

Since all commands received by RCSO include on and off delays, EPW is calculated by RCSO as

$$EPW = TIME_{OFF} - TIME_{ON} + DONOFF$$

where

$TIME_{ON} = COMMAND_{ON} + DELAY_{ON}$

$TIME_{OFF} = COMMAND_{OFF} + DELAY_{OFF}$

$DONOFF = DELAY_{ON} - DELAY_{OFF}$

The symbols in the above equations are defined below with program symbols in brackets.

EPW - Electrical pulse width [EPW]

$TIME_{ON}$ - Time at which RCS engine actually fires [ONTIME(I)]

$TIME_{OFF}$ - Time at which RCS engine actually ceases firing [OFFTIME(I)]

$DELAY_{ON}$ - On-delay for type of engine (main or vernier) [RCSOND(J)]

$DELAY_{OFF}$ - Off-delay for type of engine [RCSOFD(J)]

DONOFF - Difference between on and off delays [DONOFD(J)]

If an update is requested at some point before an engine is turned off, then the mass consumed up to that point is calculated with EPW defined as

$$EPW = TIME_{UP} - TIME_{ON} + DELAY_{ON}$$

where $TIME_{UP}$ = Time at which update is requested [AVTIME].

The fuel mass consumed by an engine firing is calculated as

$$\Delta M_{FUEL} = \frac{IMP_T}{IMP_S} + M_{LOSS}$$

where

$IMP_T = (EPW - DELAY_{ON})THR$

$IMP_S = f(EPW)$

The symbols in the above equations are defined below with program symbols in brackets.

ΔM_{FUEL} - Delta fuel mass consumed by an engine, this firing as of this update [ΔFUEL]

IMP_T - Total impulse as a function of EPW [TIMP]

IMP_S - Variable specific impulse [SPI] as a function of EPW found by evaluating a curve-fitted polynomial f(EPW)[SPITBL(K,J)](see table 4-IV). For steady state firings, i.e., EPW $\geq$ 1.0 second, the variable specific impulse is set to its steady state value taken from the input table SIMP. $IMP_S = SSIMP_S$ [SSMSPI(K)]

M_{LOSS} - Fuel loss per firing (dribble penalty, etc.) for each type of engine (main or vernier) refiled in kg from FUELI [FUELOS(J)]

THR - Total centerline vacuum thrust for each type of engine (main or vernier), in kilograms [THRK(J)]

TABLE 4-IV.- SPIMP

Variable specific impulse as a function of electrical pulse width for main and vernier engines.

EPW (seconds)		0.04	0.16	0.32	0.52	0.64	0.80	0.88	1.00
Variable specific impulse	Main	185.0	266.0	279.0	285.0	286.5	288.0	288.5	289.0
	Verniers	240.0	256.0	262.0	266.7	268.7	271.0	271.8	272.0

The total fuel mass consumed by all engines is calculated as

$$M_{FUEL} = M_{FUEL} - M_{FLAST_i} + \Delta M_{FUEL_i}$$

where

M_{FUEL} - Total fuel mass consumed by all firings of all engines as of this update [SFUEL]

M_{FLAST_i} - Mass of fuel consumed by jet i during this firing prior to this update [FUELST(I)]

ΔM_{FUEL_i} - Mass of fuel consumed by jet i for this firing as of this update [DFUEL]

The total fuel mass consumed by each engine over all its firings is maintained [FUEL(I)]. Total on-times for all engines [SONT] and for each engine [RCSONT(I)] are also maintained.

In computing forces and moments for the aft main RCS engines, vacuum impingement effects are included (ref. 16). The aft RCS vacuum impingement force increments (ΔN, ΔY, ΔA) (see table 4-V) are computed using coefficients (N_A, N_Y, N_N) (see table 4-VI) initially defined in the F frame. During initialization, these coefficients are converted to the V frame and the force increments are calculated as

$$\Delta N = N_N \cdot THR$$

TABLE 4-V.- RCS FORCE IMPINGEMENT INCREMENT

[Newtons]

JETPOD	X-axis	Y-axis	Z-axis
1	131.57839	321.20608	.00000
2	131.57839	-321.20608	.00000
3	638.54221	247.67698	1180.33559
4	638.54221	-247.67698	1180.33559
5	7.73991	.00000	61.91924
6	7.73991	.00000	61.91924
7	.00000	.00000	.00000

TABLE 4-VI.- RCS FORCE PLUME IMPINGEMENT EFFECT COEFFICIENTS

JETPOD	X	Y	Z	(Fabrication frame)
1	-0.034	0.083	0.0	
2	-0.034	-0.083	0.0	
3	-0.165	0.064	-0.305	
4	-0.165	-0.064	-0.305	
5	-0.002	0.0	-0.016	
6	-0.002	0.0	-0.016	
7	0.0	0.0	0.0	

$$\Delta Y = N_Y \cdot THR$$

$$\Delta A = N_A \cdot THR$$

The symbols used in the above equations are defined below with program symbols in brackets.

ΔN - Normal component (along z-axis) of force increment [RCSFII(J,3)]

ΔY - Side component (along y-axis) of force increment [RCSFII(J,2)]

ΔA - Axial component (along x-axis) of force increment [RCSFII(J,1)]

N_N - Normal component of impingement force coefficient [RCSFII(J,3)]

N_Y - Side component of impingement force coefficient [RCSFII(J,2)]

N_A - Axial component of impingement force coefficient [RCSFII(J,1)]

THR - Centerline vacuum thrust of aft RCS engine in newtons [THRN(1)]

Data for the aft RCS vacuum impingement moment coefficients (N_ℓ, N_m, N_n) (see table 4-VII) were derived separately and were initially defined in the V frame, thus no conversion is necessary. During initialization, the moment increments $(\Delta N_\ell, \Delta N_m, \Delta N_n)$ (see table 4-VIII) are calculated as

$$\Delta N_\ell = N_\ell \cdot A_{AVG_\ell} \cdot THR$$

$$\Delta N_m = N_m \cdot A_{AVG_m} \cdot THR$$

$$\Delta N_n = N_n \cdot A_{AVG_n} \cdot THR$$

where

ΔN_ℓ - Roll component of impingement moment increment [RCSTII(I,1)]

ΔN_m - Pitch component of impingement moment increment [RCSTII(I,2)]

TABLE 4-VII.- RCS TORQUE IMPINGEMENT INCREMENT

[Newton-meters]

JETPOD	X-axis	Y-axis	Z-axis
1	2393.58603	-271.24019	-3308.55228
2	-2393.58603	-271.24019	3308.55228
3	-3447.64221	12399.55164	-1654.27614
4	3447.64221	12399.55164	1654.27614
5	614.86613	503.73178	-157.55011
6	-614.86613	503.73178	157.55011
7	.00000	.00000	.00000

Note: These forces and torques due to plume impingement effects are added directly to the forces and torques of each individual jet in the appropriate jet pod.

TABLE 4-VIII.- RCS TORQUE PLUME IMPINGEMENT EFFECT COEFFICIENTS

JETPOD	X	Y	Z	(Vehicle frame)
1	0.218	-0.007	-0.084	
2	-0.218	-0.007	0.084	
3	-0.314	0.320	-0.042	
4	0.314	0.320	0.042	
5	0.056	0.013	-0.004	
6	-0.056	0.013	0.004	
7	0.0	0.0	0.0	

ΔN_n - Yaw component of impingement moment increment [RCSTII(I,3)]

N_ℓ - Roll component of impingement moment coefficient [RCSIIT(I,1)]

N_m - Pitch component of impingement moment coefficient [RCSIIT(I,2)]

N_n - Yaw component of impingement moment coefficient [RCSIIT(I,3)]

A_{AVG_ℓ} - Roll component of average moment arm for aft RCS engines [ARM(1)]

A_{AVG_m} - Pitch component of average moment arm for aft RCS engines [ARM(2)]

A_{AVG_n} - Yaw component of average moment arm for aft RCS engines [ARM(3)]

Individual RCS nominal forces [RCSFI(J)] (without impingement effects) are input via DATA cards and converted to newtons. Individual RCS nominal torques [RCSTOR(J)] are calculated at initialization. Total forces [RRCSFJ(J)] and torques [RRCSTJ(J)] for each jet are then computed as the sum of the jet's nominal force or torque and the impingement force or torque increment. It should be noted that the values for the impingement increments apply to one jet, i.e., the increments are those created by the firing of one RCS aft main engine, but they will always be equal for all the engines in a particular group. Thus, these increments are calculated for groups but added to the forces and moments of individual jets.

Total forces [SRCSF(I)] and total torques [SRCST(I)] generated by firing RCS engines are the sums of individual forces and torques over all engines which are on. If, during a simulation run, the mass properties of the vehicle change, the moment arms, impingement moment increments, nominal moments and the individual total moments are recalculated.

It is possible to exclude the impingement effects from all force and moment calculations by changing the input value of a flag called IMPFLG (Location 1225 in the RCS COMMON block). Normally, IMPFLG is defined as one and impingement effects are included. By changing its value to zero, no impingement effects will be incorporated.

INPUT/OUTPUT

This model has no calling arguments. However, values for three parameters are obtained from the mass properties routine for its initialization. They are:

$\underline{R}_{VX}$ - Position vector of the vehicle's center of mass (X) in the vehicle (V) body frame (meters). [RVX]

$\underline{R}_{FV}$ - Position vector of the vehicle (V) body frame in the fabrication (F) frame (meters). [RFV]

$^{V}M_{F}$ - Fabrication (F) to vehicle (V) body frame attitude matrix. [AMFV]

All other data values necessary for this model's computations are contained in internal data statements (see ref. 13) which can be changed using the SOAP data input scheme. The input parameters to this model are:

$\underline{R}_{FJ_i}$ - Positions (stations) of the ACPS engines (J) in the fabrication (F) frame; input units are inches, which are converted to meters for simulator calculations. [RCSSTI] (See table 4-II.)

$\underline{Q}_i$ - Force vectors of the ACPS engines in the vehicle body frame; input units are pounds, which are converted to newtons for simulator calculations. [RCSFI] (See table 4-II.)

AXIS - Hollerith array (used for output print only), which contains the force direction for each engine, e.g., 4H - X for engine 1, 4H + Y for engine 5, etc.

PCSOND - jth size ACPS engine on-delay (0.016,0.010,0.0 sec)

RCSOFD - jth size ACPS engine off-delay (0.0,0.0,0.0 sec)

TMINPL - jth size ACPS engine minimum impulse time (0.04,0.04,0.0 sec)

FUELI - Mass loss per jth size ACPS engine firing; input units are pounds, which are converted to kilograms for simulator calculations and output. (0.0,0.0,0.0 lb)

JSCFLG - ACPS engine size change flag. This array identifies all ACPS engines with a particular size engine group (see table 4-II).

NJ - Number of ACPS engines that are to be used; no units. (NJ = 46)

All of the above parameters may be changed by input.

During initialization, six parameters are calculated:

$\underline{R}_{VJ_i}$ - Positions of J in the V body frame. [RCSPOS] (m)

$\underline{N}_i$ - Torque vectors of the ACPS engines [RCSTOR]. This will remain constant if the center of mass of the vehicle is not changing. (N·m)

RCSFII - Force impingement increment for each jet group

RCSTII - Torque impingement increment for each jet group

RRCSFJ - Resultant or total force for each jet (sum of RCSFOR for jet j and RCSFII for the jet group of jet j)

RRCSTJ - Resultant or total torque for each jet (sum of RCSTOR for jet j and RCSTII for the jet group of jet j)

The values input and calculated above are printed out during initialization. Table 4-II shows the values of JSCFLG, JETPOD, $\underline{R}_{F,J_i}$, and $\underline{Q}_i$. Also printed at initialization are the steady state and minimum impulse values for the variable specific impulse.

When an ACPS engine(s) is turned on (1) or off (0), optional print is available. This print is controlled by a print flag (RCSPC). When RCSPC is 0.0, no print is received when commands are issued. When RCSPC is 1.0, the following information is printed:

1. Time of this ACPS action
2. The number of ACPS engines on and which ones they are

3. The command being issued (501 through 540)
4. The axis along which the force is directed
5. The state (on or off) of the command
6. The number of times this ACPS engine has been fired
7. The total on-time for this ACPS engine
8. The fuel used by this ACPS engine
9. The total fuel used by all ACPS engines
10. The force vector for this ACPS engine
11. The torque vector for this ACPS engine
12. The resultant or total force vector for all ACPS engines which are on
13. The resultant or total torque vector for all ACPS engines which are on

When a jet command is incorporated, the vehicle program updates its dynamics. A vehicle dynamics print flag, PAVRDC, can be set to force the vehicle to print its response to the jet command(s). PAVRDC is defined initially by the vehicle model to be -1.0 via a data statement and is in the vehicle COMMON block (location 131). PAVRDC has three settings: -1.0, 0.0, and 1.0 If equal to -1.0, no print will be obtained and the RCS model cannot reset it. The value of PAVRDC can only be changed from -1.0 by data input. When PAVRDC is equal to 0.0, this RCS model will set it to 1.0 indicating to the vehicle model to print its dynamics response to a command. The vehicle model resets PAVRDC to 0.0 for future commands.

RSCO uses two external subroutines, POLY and PEVAL. At initialization, POLY is called to construct a polynomial curve to a set of data points for variable specific impulse. PEVAL is an associated routine which evaluates the polynomial for a given electrical pulse width.

SOURCE

The RCS0 subroutine is essentially identical to RCS13 with the following exceptions.

1. The incorporation of RCS vacuum impingement force and torque increments
2. The incorporation of a variable specific impulse for fuel calculations
3. New data on jet positions, forces, on and off delays, minimum impulse times, and mass loss per firing

Data for ACPS engine locations and thrust direction were taken from table 2.4.1.3 of reference 15. This data is for the RI Orbiter Vehicle 5 as shown in Rockwell drawing number VC 70-000002A in reference 13. The jet numbering system is from reference 14. The specific impulse data is from reference 15. The impingement effect data is from reference 16. The subroutines POLY and PEVAL are described in references 17 and 18, respectively.

PROGRAM VERIFICATION

The RCS0 subroutine has been verified with closed-loop SOAP runs and the printed output agrees with hand calculations.

RCS

Subroutine names RCS1, RCS2.

PROGRAM DESCRIPTION

These RCS models compute the force, torque, and fuel use which result from commands issued to the Reaction Control System by the flight control routines. The RCS routines call the appropriate mass properties model to initiate update of the inertia matrix. The various Subroutine names supplied allow the SOAP to distinguish among the models which correspond to various vehicles, but denote no change in algorithm.

The RCS models maintain a COMMON block with location, thrust level, thrust direction, fuel container, etc., for each of up to 50 jets. Commands to individual jets are issued by the flight control system along with time tags for the commands. Each RCS model responds only to those commands designated for it and ignores commands for other vehicles.

Upon receipt of a command to either turn a jet on or off, the RCS model

a. Delays the action by an on or off-delay if desired.

b. Delays off-commands until a jet has been on for some minimum time.

c. Computes the thrust due to each jet. If desired, the jet response can be modeled as having a specific impulse which is a function of time. The routine POLY (or DPOLY) is used to generate the polynomial describing the variation of specific impulse and PEVAL (or DPEVAL) evaluates the impulse.

d. Computes the fuel use associated with this jet firing. This computation may include the effects of the varying specific impulse noted above (so fuel use is not simply proportional to time on) and may include a loss of fuel by the jet. This loss is modeled as a fixed amount per firing.

e. Causes the mass properties of the vehicle to be updated.

f. Computes the torque due to each jet and the total torque about the current center of mass of the vehicle.

g. Axiliary, nondynamic quantities such as which jets are on, how long each is on, how many times each has been used, etc., are computed.

As currently configured, RCS1 accepts commands numbered 551 → 551 + n and RCS2 accepts commands 601 → 651 + m where n and m are the number of jets specified to be in use.

INPUT/OUTPUT

RCS has no calling arguments.

Prog. Symbol	I/O	COMMON block name	COMMON block location	Dimen.	Description (unit)
PAVRDC	O	ACTV#C	131	1	Print flag
AVTIME	I	ACTV#C	138	1	Active vehicle time (sec)
INIT	I	ENVCOM	322	1	Initialization flag
NCCMDS	I	ENVCOM	323	1	Number of commands on push list
TPUSH	I	ENVCOM	431	50	Time associated with command in CMMD (sec)
CMMD	I	ENVCOM	481	50	Command identifier
DATI	I	ENVCOM	531	50	Command, 1 = turn jet on, 0 = turn jet off
LCBRCS	I*	LECCOM	1713	1	Length of RCS COMMON block
RVX	I	MASP#C	2	3	Center of mass location in body fixed coordinates (m)
SRCSF	O	RCS#C	1	3	Sum of RCS jet forces (N)
SRCST	O	RCS#C	4	3	Sum of RCS jet torques about vehicle center of mass (N·m)
SFUEL	O	RCS#C	7	1	Sum of RCS jet fuel used (kg)
FUEL	O	RCS#C	8	50	Fuel used by each jet (kg)
NJ	I*	RCS#C	58	1	Number of jets present
RCSDT	O	RCS#C	59	1	Time to which RCS is updated
RCSSTA	I*	RCS#C	61	3xNJ (But $\leq$150)	RCS jet locations in body coordinates. The order is x, y, z for jet 1, x, y, z for jet 2, etc., (in)
RCSPOS	O	RCS#C	211	150	RCS jet locations (m)
RRCSFJ	O	RCS#C	361	150	Forces due to RCS jets individually (N)

*Values which require initialization, either by the user or by default

#A symbol which depends on the vehicle involved

Prog Symbol	I/O	COMMON block name	COMMON block location	Dimen.	Description (unit)
RRCSTJ	O	RCS#C	511	150	Torques due to RCS jets individually (N•m)
CRCSS	O	RCS#C	661	50	Commanded RCS jet states, 0 = off, 1 = on
PRCSS	O	RCS#C	711	50	Present RCS jet states, 0 = off, 1 = on
RCSPF	O	RCS#C	761	50	RCS print flag
ONTIME	O	RCS#C	811	50	Time RCS jet is commanded on (sec)
RCSONT	O	RCS#C	861	50	Time for which RCS has been on
FIREN	O	RCS#C	911	50	Number of times a jet has been fired
TMINPL	I*	RCS#C	961	3	Minimum time for jet to be on, by categories (sec)
RCSFMG	I*	RCS#C	964	1	RCS jet force magnitude
SONT	O	RCS#C	965	1	Sum of the RCS jet firing times (sec)
RCSFLG	O	RCS#C	966	1	RCS jet state change flag
RCSPC	O	RCS#C	967	1	RCS print control
IRCSS	O	RCS#C	968	1	Number of jets on
IRCS	O	RCS#C	969	1	Jet on indicator
AXIS	I*	RCS#C	1019	50	Direction of RCS jet force, literals used for prints only
FUELI	I*	RCS#C	1071	3	Fuel lost per jet firing, three categories (kg)
RCSOND	I*	RCS#C	1074	3	Delay between jet-on command and jet firing, three categories (sec)
RCSOFD	I*	RCS#C	1077	3	Delay between jet-off command and actual jet turn off, three categories (sec)

Prog Symbol	I/O	COMMON block name	COMMON block location	Dimen.	Description (unit)
DONDFD	I*	RCS#C	1080	3	Difference between on and off delays (sec)
JSCFLG	I*	RCS#C	1083	50	Jet size change flag
JETPOD	I*	RCS#C	1133	50	Jet pod membership numbers, identify fuel source for each jet
RCSFI	I*/O	RCS#C	1226	150	RCS jet force vectors in the order x, y, z force due to jet 1; x, y, z force due to jet 2, etc. (Input in lb, changed to N internally)
RCSTOR	O	RCS#C	1376	150	RCS jet torques (N•m)
SPIMP	I*	RCS#C	1526	16	Table of variable specific impulses (sec)
CFFTIME	O	RCS#C	1542	50	Time at which jets are commanded off (sec)
SPI	O	RCS#C	1592	1	Specific impulse (sec)
EPWTBL	I*	RCS#C	1593	8	Electrical pulse width table (sec)
DFUEL	O	RCS#C	1601	1	Change in fuel used by a jet this firing
EPW	O	RCS#C	1602	1	Electrical pulse width
TIMP	O	RCS#C	1603	1	Total impulse (kg/s)
THRNI	I*	RCS#C	1604	2	Centerline vacuum thrust of jets, two types (lb)
FUELST	O	RCS#C	1606	50	Fuel used by each jet in this firing prior to this update
THRK	O	RCS#C	1659	2	Centerline vacuum thrust (N)
SSMSPI	O	RCS#C	1661	4	Steady state and minimum impulse values for specific impulse

4.9 SENSOR CONTROL

SNSC3

PROGRAM DESCRIPTION

SNSC3 determines the times at which the sensor models are called and calls each active model at the appropriate time. Optional printout of these times may be selected if desired.

MATH MODEL

At initialization, all sensor model update times are set equal to the time to which the active vehicle is to update. The update times for those models which are not active are set to a very large number on the second pass initialization. During a normal update, the value stored as the current sensor control time, t_s, is the next most current update time of either the vehicle or a sensor model. t_s is then compared with the update times for each of the active sensor models to see if the time has been reached for that particular model to be called, i.e.,

$$t_s \approx tn_m$$

where tn_m is the sensor model update time. The comparisons are repeated every update until the time which has been set for a model to be called or started is reached. When this time has been reached, the model is called and the time for the next call is determined as follows:

$$t_m = tn_m$$

$$tn_m' = tn_m + \Delta t_m$$

where

t_m is the current time at which the model is being called

tn_m is the model update time

tn_m is the next time at which the model is to be called, or the next update time

Δt_m is the model's maximum update interval

The two times calculated above may be printed after each update.

INPUT/OUTPUT

SNSC3 uses three print control flags. If SNSPC is 1.0, a test is made to see if it is time to print and if the print time, SNSPT, should be updated. If it is time to print and the print update interval, SNSPDT, is greater than 0.0, SNSPT is updated before the next print.

SNSC3 requires as input from CONST

ROCDEL Small number used to check for roundoff

FINITY Large number used if sensor model activity switch is off

The model activity switches and model update intervals for all of the sensor models called come from ENVCOM. These models are RADALT, BARALT, LNDAID, RNGNAV, and AIRDAT.

PROGRAM VERIFICATION

SNSC3 was checked out in open-loop simulations.

5. FLIGHT CONTROL SYSTEMS

The flight control system (also termed flight software) models the activities of each vehicle and pilot in response to both the environmental conditions and to some overall mission plan. Functions which are elsewhere termed guidance, navigation, and control are all included in this category.

The flight control routines are arranged into blocks which model a specific vehicle. Typically, these blocks interact mainly within themselves. Information about the outside world can be obtained from the environment models, the pilot model, or from sensor models. Communication within a flight control system is typically handled by a COMMON block whose name begins with the letters FS. The structure of these blocks is determined by the vehicle involved.

Flight control routines are either called (by other flight control routines within the same vehicle) or are scheduled. In the latter case, the SOAP executive calls each routine at the times specified. Scheduling can be performed at the outset of the simulation by input or can be performed by other routines during the course of the simulation.

The flight control routine documentation is arranged by vehicle. It includes a general description of the operation of the routines which are specific to a vehicle. Pilot models and utility routines are excluded.

The flight control system models available include the Maneuverable Television (MTV), the Detached Equipment Carrier (DEC), and the Space Shuttle Orbiter (SSO).

5.1 MTV AND DEC (MMU TYPE)

These vehicles are based on the control system of the Manned Maneuvering Unit described in reference 19. A block diagram of the operation of this control system is shown in figure 5-1.

The pilot (a user-supplied routine) issues commands to the hand controllers and/or the attitude hold button. These commands are carried in COMMON block locations to MTVFSW, the main control system routine. MTVFSW in turn calls the phase plane routine PPMTV, which may issue rotational commands to either halt the vehicle or to move it back to zero attitude.

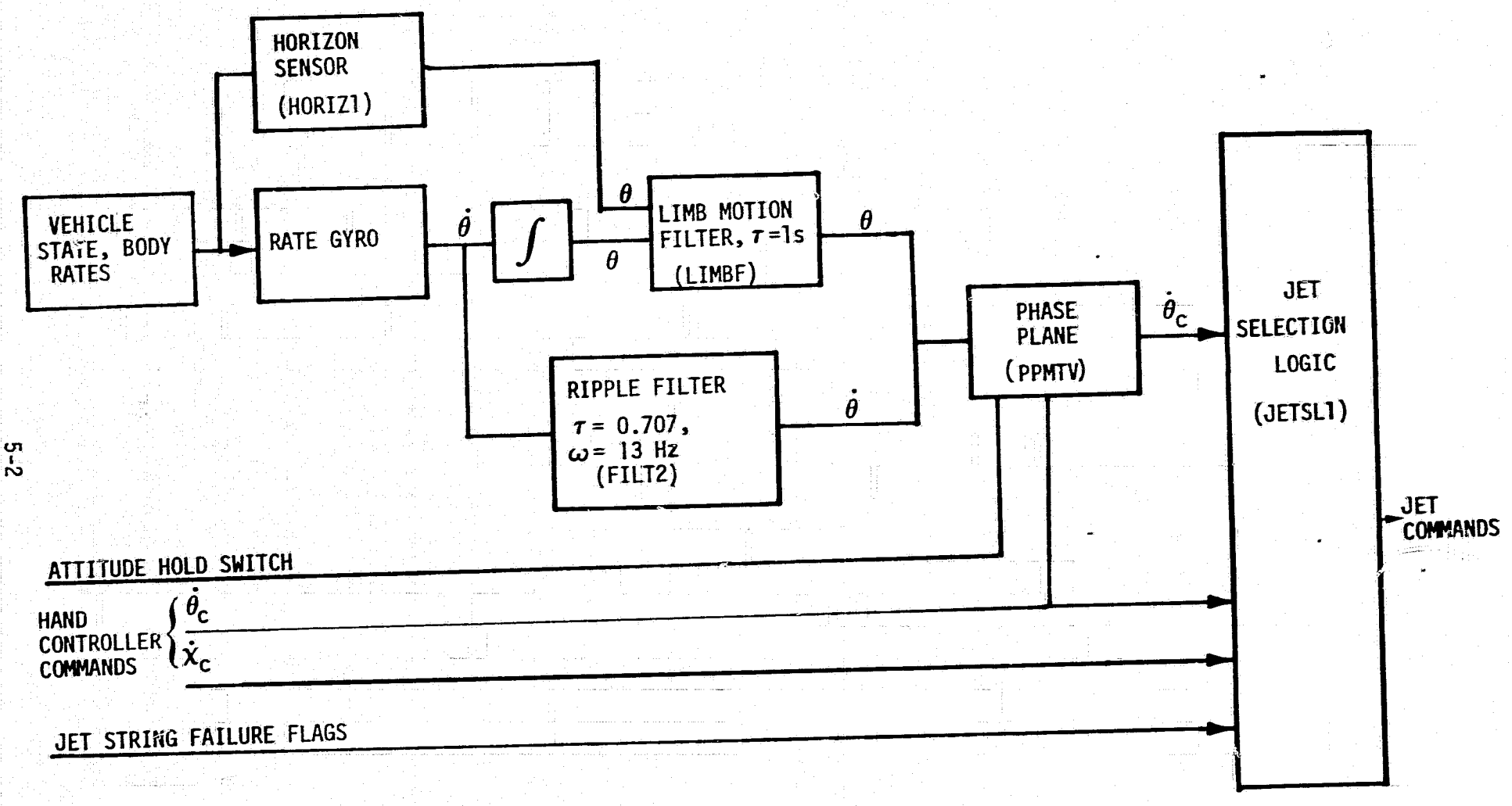


Figure 5-1.- MTV/DEC control system.

The rotational and translational commands are then combined by MTVFSW and sent to the routine JETSL where the appropriate choice of jets to fire is made. These commands are then filed by MTVFSW with the SOAP command executive.

The operation of the jet selection logic is simply an implementation of the selections shown in table 5-I. The jet locations and numbers are shown in figure 5-2. The properties of the jets, their locations, thrust levels, etc., are not known to the flight control system. They are specified as desired by the user. The jet numbers and locations are indicative of the situation for which the control system was designed. Table 5-I displays the jets which are chosen in response to commands or combinations of commands. The switch between prime mode and backup modes is initiated by setting jet string failure flags.

Figure 5-3 displays the phase plane logic. In the region indicated by "Thrusters pulsing" the flight control commands a 0.008-second rotational command and then waits 0.32 second before performing any further action. There is a separate phase plane for each axis.

The rates and attitudes which are used by the phase plane are body rates and attitudes. The attitudes are obtained by integrating the body fixed rates (from environment) and then filtering these rates. Two filters are applied. The limb motion filter (modeled in LIMBF) was designed to prevent the vehicle from responding to bodily contortions of the astronaut using the manned maneuvering unit. This is a first-order attitude filter with a time constant of 1 second. There is also a second-order filter with ω = 13.Hz and ζ = 0.707 applied to the sensed rates. This is termed the ripple filter and is modeled in FILT2. The attitude associated with any body axis is initialized after attitude hold is initiated when the attitude rate drops below 0.2 deg/sec. The attitude hold on an axis is released either when a rotation about that axis is commanded or when attitude hold is turned off.

The Detached Equipment Carrier control system differs in that the pitch and roll attitudes received by the phase plane are generated by a horizon sensor. The pitch rate which the phase plane receives is biased by a constant equal to the negative value of the orbital angular rate (i.e., 2π radians per period). Yaw is controlled by a rate gyro. The deadband limits

TABLE 5-I.- JET SELECTIONS

(a) X, theta (pitch), psi (yaw) logic, prime

No.	Command	Prime mode							
	X θ ψ	12	7	10	19	1	16	13	4
1	+	1	1	1	1				
2	-					1	1	1	1
3	+			1		1			
4	-	1						1	
5	+		1			1			
6	-	1					1		
7	+ +			1	1				
8	+ -	1	1						
9	- +					1	1		
10	- -							1	1
11	+ +		1		1				
12	+ -	1		1					
13	- +					1		1	
14	- -						1		1
15	+ +				1	1			
16	+ -			1			1		
17	- +		1					1	
18	- -	1							1
19	+ + +				1				
20	+ + -			1					
21	+ - +		1						
22	+ - -	1							
23	- + +					1			
24	- + -						1		
25	- - +							1	
26	- - -								1

TABLE 5-I.- CONTINUED

(b) Y, phi (roll), psi (yaw) logic, prime

No.	Command	Prime mode							
	Y φ Ψ	8	17	5	20	2	23	11	14
27	+	1	1	1	1				
28	-					1	1	1	1
29	+	1						1	
30	-				1		1		
31	+	All Zero							
32	-								
33	+ +	1	1						
34	+ -			1	1				
35	- +							1	1
36	- -					1	1		
37	+ +		1	1					
38	+ -	1			1				
39	- +						1	1	
40	- -					1			1
41	+ +		1					1	
42	+ -	1							1
43	- +			1			1		
44	- -				1	1			
45	+ + +		1						
46	+ + -	1							
47	+ - +			1					
48	+ - -				1				
49	- + +							1	
50	- + -								1
51	- - +						1		
52	- - -					1			

TABLE 5-I.- CONTINUED

(c) Z, phi (roll), theta (pitch) logic, prime

No.	Command	Prime mode							
	Z ϕ θ	3	24	9	18	12	15	6	21
53	+	1	1	1	1				
54	-					1	1	1	1
55	+	All Zero							
56	-								
57	+	All Zero							
58	-								
59	+ +	1		1					
60	+ -		1		1				
61	- +					1		1	
62	- -						1		1
63	+ +		1	1					
64	+ -	1			1				
65	- +						1	1	
66	- -					1			1
67	+ +	All Zero							
68	+ -								
69	- +	All Zero							
70	- -								
71	+ + +		1						
72	+ + -	1							
73	+ - +			1					
74	+ - -				1				
75	- + +							1	
76	- + -								1
77	- - +						1		
78	- - -					1			

TABLE 5-I.- CONTINUED

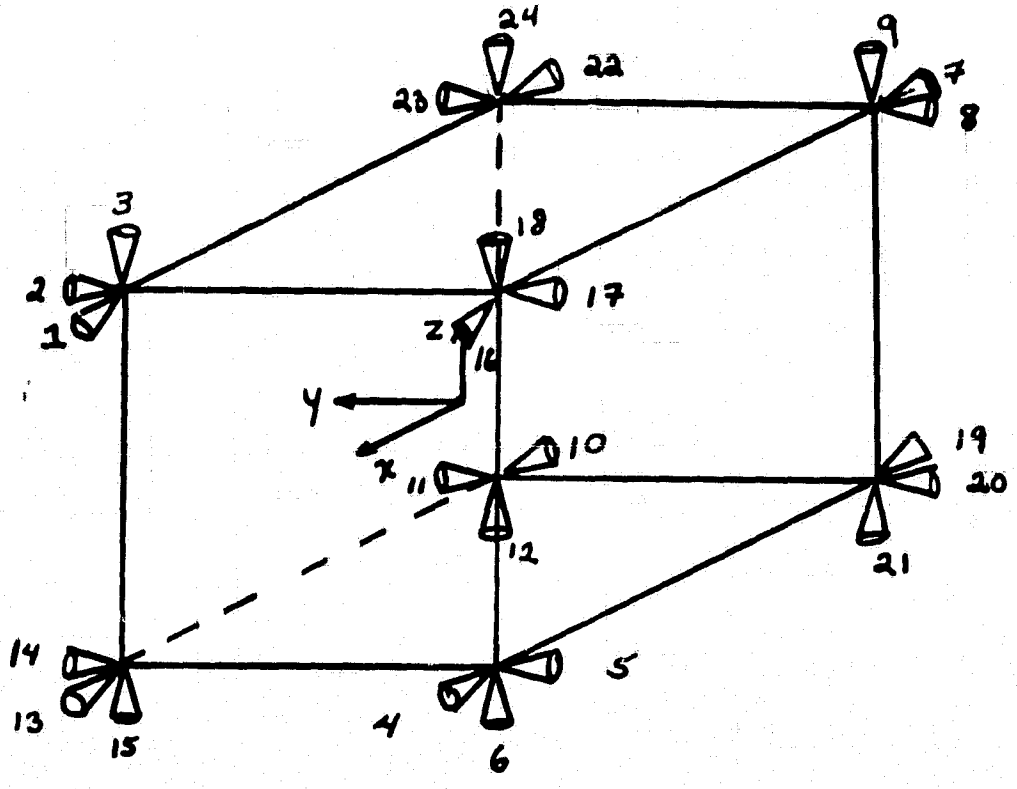
(d) X, theta (pitch), psi (yaw) logic, backup

No.	Command	Backup mode A				Backup mode B			
	X θ ψ	7	10	1	4	22	19	16	13
1	0 0 0								
2	+ 0 0	1	1			1	1		
3	- 0 0			1	1			1	1
4	0 + 0		1	1			1	1	
5	0 - 0	1			1	1			1
6	0 0 +	1		1			1		1
7	0 0 -		1		1	1		1	
8	+ + 0		1	1			1	1	
9	+ - 0	1			1	1			1
10	- + 0		1	1			1	1	
11	- - 0	1			1	1			1
12	+ 0 +	1		1			1		1
13	+ 0 -		1		1	1		1	
14	- 0 +	1		1			1		1
15	- 0 -		1		1	1		1	
16	0 + +			1			1		
17	0 + -		1					1	
18	0 - +	1							1
19	0 - -				1	1			
20	+ + +	1	1	1			1		
21	+ + -		1			1	1	1	
22	+ - +	1				1	1		1
23	+ - -	1	1		1	1			
24	- + +			1			1	1	1
25	- + -		1	1	1			1	
26	- - +	1		1	1				1
27	- - -				1	1		1	1

TABLE 5-I.- CONCLUDED

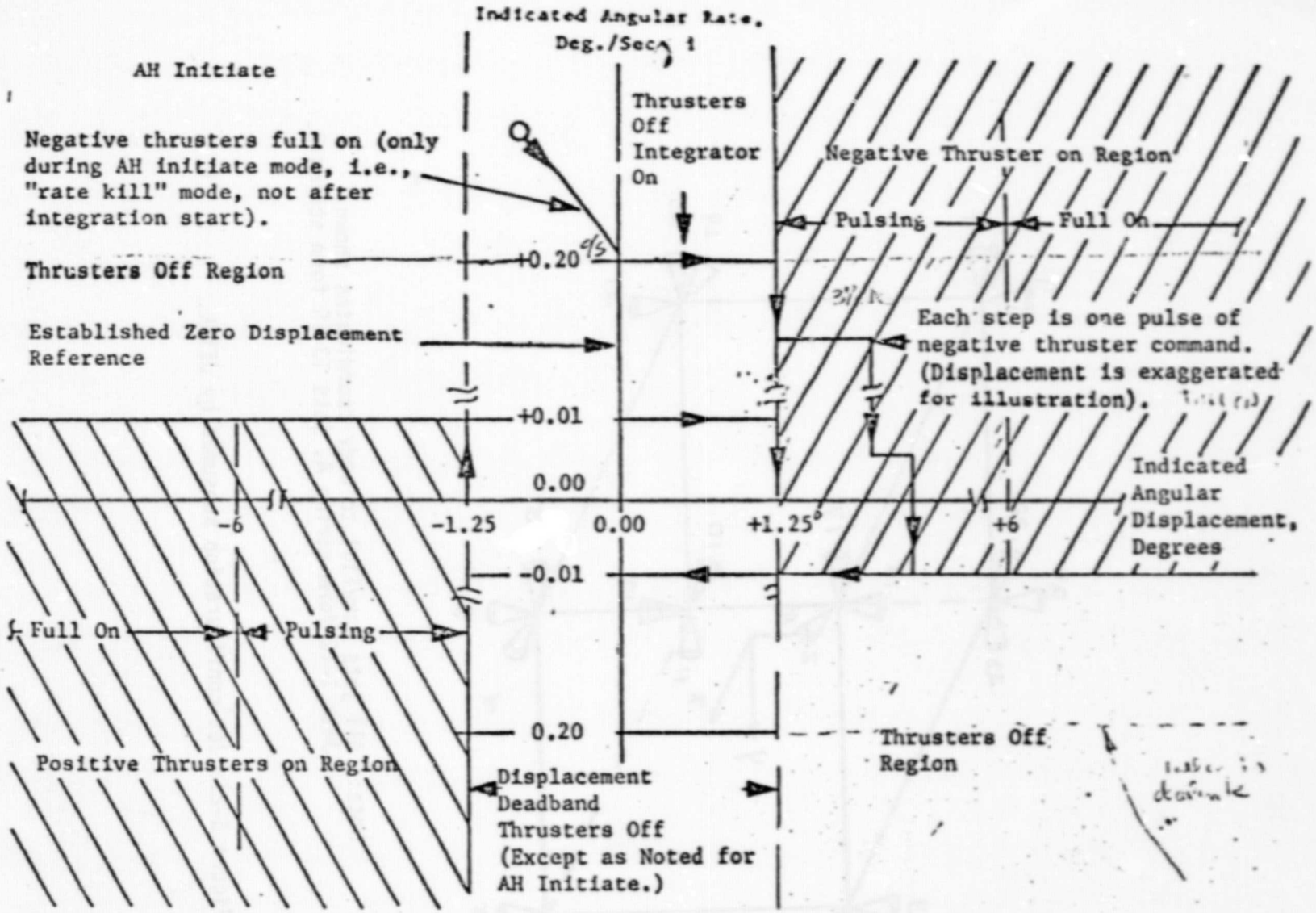
(e) Y, Z, phi (roll) logic, backup

No	Cmd	Backup mode A								Backup mode B							
	Y Z φ	8	5	2	11	3	9	12	6	17	20	23	14	24	18	15	21
28	0 0 0																
29	+ 0 0	1	1							1	1						
30	- 0 0			1	1							1	1				
31	0 + 0					1	1							1	1		
32	0 - 0							1	1							1	1
33	0 0 +	1			1					1			1				
34	0 0 -		1	1							1	1					
35	+ + 0		1			1					1			1			
36	+ - 0	1						1		1						1	
37	- + 0				1		1						1		1		
38	- - 0			1					1			1					1
39	+ 0 +	1			1					1			1				
40	+ 0 -		1	1							1	1					
41	- 0 +	1			1					1			1				
42	- 0 -		1	1							1	1					
43	0 + +	1			1					1			1				
44	0 + -		1	1							1	1					
45	0 - +	1			1					1			1				
46	0 - -		1	1							1	1					
47	+ + +	1			1					1			1				
48	+ + -		1	1							1	1					
49	+ - +	1			1					1			1				
50	+ - -		1	1							1	1					
51	- + +	1			1					1			1				
52	- + -		1	1							1	1					
53	- - +	1			1					1			1				
54	- - -		1	1							1	1					



Note: All jets parallel to body coordinates shown.
Jets 1-12 form string A, jets 13-19 form string B.

Figure 5-2.- Jet configuration subsumed by JETSL.



Typical convergence to limit cycle when AH is initiated while MMU rate is > + 0.2°/sec. All values are nominal (for illustration only). Effect of system delays not shown. Sketch represents each axis.

Figure 5-3.- Limit cycle.

are changed to 1° and 5°. Pilot commands can initiate or terminate the use of the horizon sensor, inertial attitude hold, and free attitude mode.

A diagram of the control system implementation is shown in figure 5-1. Subroutine names are in parentheses. The routines MTVFSW and RCALSW are transparent to the user but also exist as part of the flight control system. The map instructions and control statements required to obtain a simulation using these flight control systems are available in system files as outlined in section 2.

INPUT/OUTPUT

All input/output is via the COMMON blocks FS1C and FS2C. Interfaces with the pilot are defined as follows:

Prog. Symbol	I/O	COMMON block name	COMMON block location	Dimen.	Description (units)
WV	I	ACTV#C	26	3	Body attitude rates in rad/sec
HDCTRL	I	FS#C	5	6	Hand controller commands for roll, pitch, yaw, x, y, z where +1 = positive -1 = negative 0 = no command
ISFL	I	FS#C	23	1	Jet string fail flag 0 = no failures 1 = string A failed 2 = string B failed
ISWON	I	FS#C	12	1	Attitude hold on flag, 1 = on
ISWOF	I	FS#C	13	1	Attitude hold off flag 1 = off
TH	I/O	FS#C	14	3	Array of body angles used by phase plane, deg
THD	I/O	FS#C	17	3	Array of body rates used by phase plane, deg/s

# Denotes 1 or 2

Prog. Symbol	I/O	COMMON block name	COMMON block location	Dimen.	Description (units)
IATHDL	0	FS#C	20	1	Last pass attitude hold flag, = 1 for first pass through phase plane after initiating attitude hold
IFGHSI	I	FS#C	70	1	Horizon sensor input flag. If 1 use horizon sensor, otherwise use gyro
NPPCMD	0	FS#C	90	3	Number of phase plane commands issued

Routines in MTV flight control systems:

Routine	Scheduled Interval	Purpose
MTVFSW	0.16s, typical	Control system main driver
FILT2	0.01s	Ripple filter
LIMBF	0.01s	Limb motion filter
PPMTV	Called by MTVFSW	Phase plane logic
JETSL1	Called by MTVFSW	Jet selection logic
RCALSW	Scheduled by MTVFSW	Produces a minimum duration jet firing

5.2 SSO

The onorbit flight software is configured to the Space Transportation System-1 (STS-1) Level C as described in reference 20. The overview of the onorbit Digital Autopilot (DAP) is presented in figure 5-4. Also included in the software package are the Attitude processor which receives Inertial Measurement Unit (IMU) attitude information for further processing and the Universal Pointing processor which generates guidance command information to be used by the autopilot. For detailed information concerning these processors and the onorbit DAP, consult reference 20. Table 5-II is a list of the major flight software routines that can be correlated to figure 5-4.

Diagrams of the operation of the DAP are given in figure 5-4 and the routines which model each block are named. Detailed descriptions of the algorithms used are available in reference 20.

In addition to the DAP functions, the Shuttle flight control system accepts input and output via the routines which model the Universal Pointing Processor (UNPTPR) and the DAP Load Select (DAPSLT).

The inputs to these routines correspond to inputs via the CRT display 2011 (figures 5-5 and 5-6) and via a control panel (C3, fig. 5-7). Reference 21 contains detailed descriptions of the switch and display settings required to cause the Shuttle to perform the desired maneuvers. On figures 5-5 and 5-6, the COMMON block locations (and in the case of fig. 5-6, the required values) which correspond to pilot inputs are noted. On figure 5-5, the functions "Land Site Update" and Vent Door Control" are displayed. These functions are not modeled by the SOAP.

In addition to the switch panel and the two CRT displays, the Shuttle flight control also accepts input from rotational and translational hand controllers. Setting these inputs is discussed in the section on user input requirements.

Communication within the Shuttle flight control system is through the COMMON block FSCOM. Definitions of the variables within this block correspond

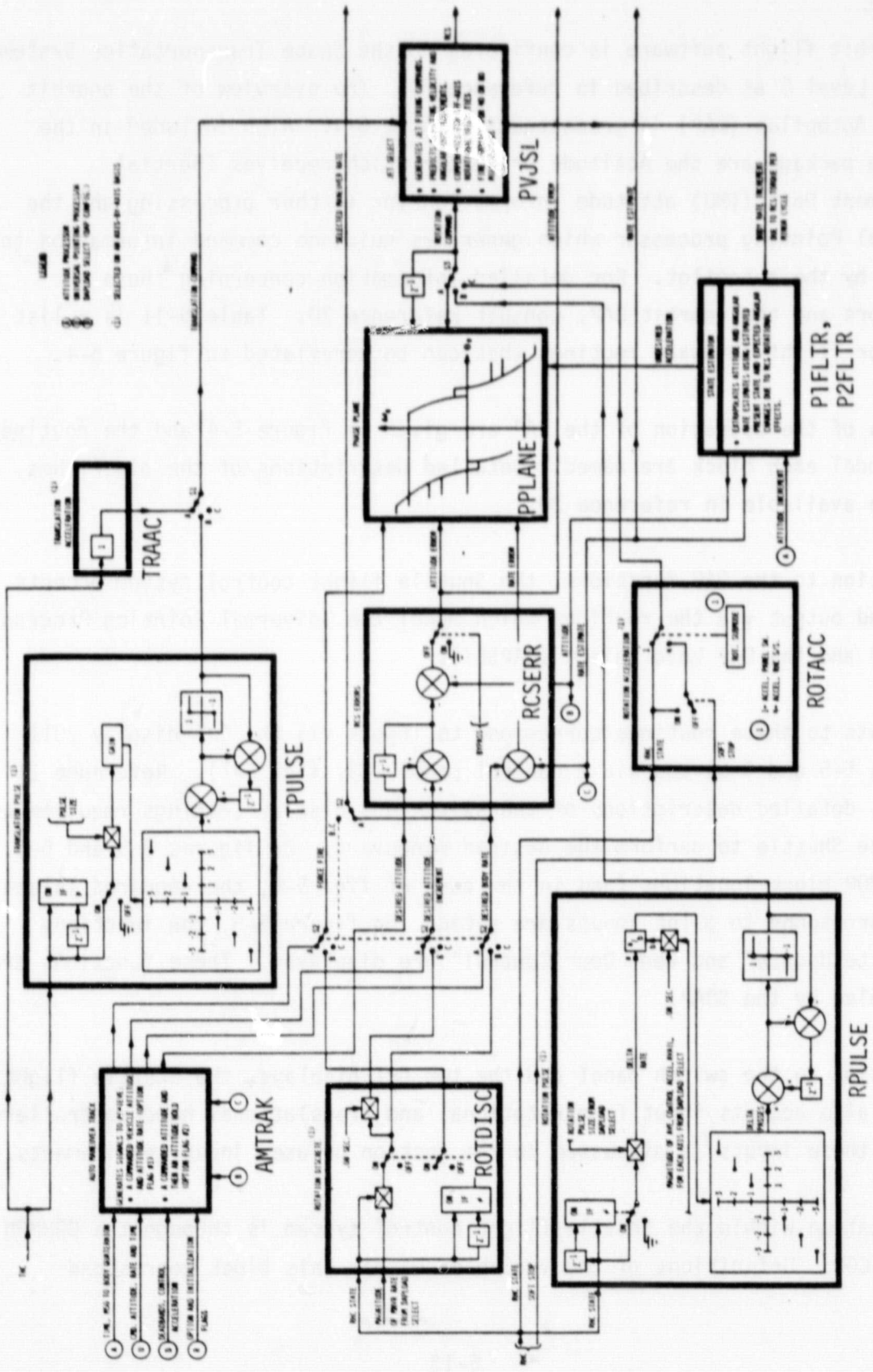


Figure 5-4.- Overview, onorbit digital autopilot.

TABLE 5-II.- FLIGHT SOFTWARE FUNCTIONS

Routine	Associated with	Function	Called by
RECON	DAP	DAP main driver	OFC
AMTRAK	DAP	Auto maneuver track	RECON (as required)
TPULSE	DAP	Translation pulse	RECON (as required)
TRAACC	DAP	Translation acceleration	RECON (as required)
RODISC	DAP	Rotation discrete	RECON (as required)
RPULSE	DAP	Rotation pulse	RECON (as required)
ROTACC	DAP	Rotational acceleration	RECON (as required)
RCSERR	DAP	RCS errors	RECON (as required)
PPLANE	DAP	Phase plane	RECON (as required)
P1FLTR	DAP	State estimator	RECON (as required)
P2FLTR	DAP	State estimator	RECON (as required)
PVJSL	DAP	Jet select	RECON (as required)
DAPSLT	I/O	DAP load select	RECON (as required)
ATTPRC	I/O	Attitude processor	Scheduled at 0.16 sec intervals
UNPTPR	I/O	Universal pointing processor	Scheduled at 1.92 sec intervals
UNPTSP	I/O	Universal pointing specialist	Scheduled at 1.92 sec intervals
OFC		Scheduled FSW driver	Scheduled at 0.08 sec intervals

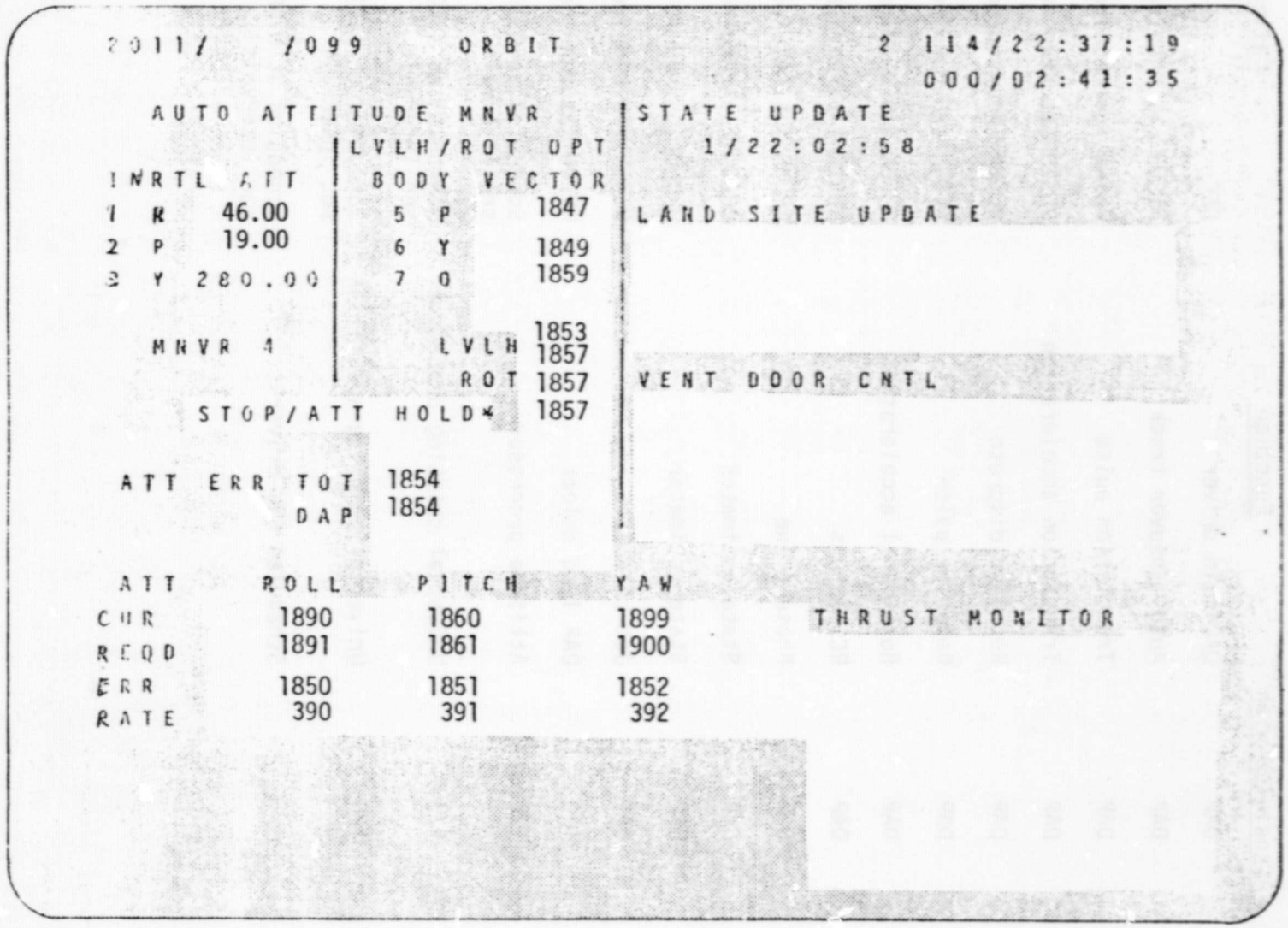


Figure 5-5.- Universal Pointing Processor display.
(Bold numbers denote location in FSCOM associated with the CRT input position)

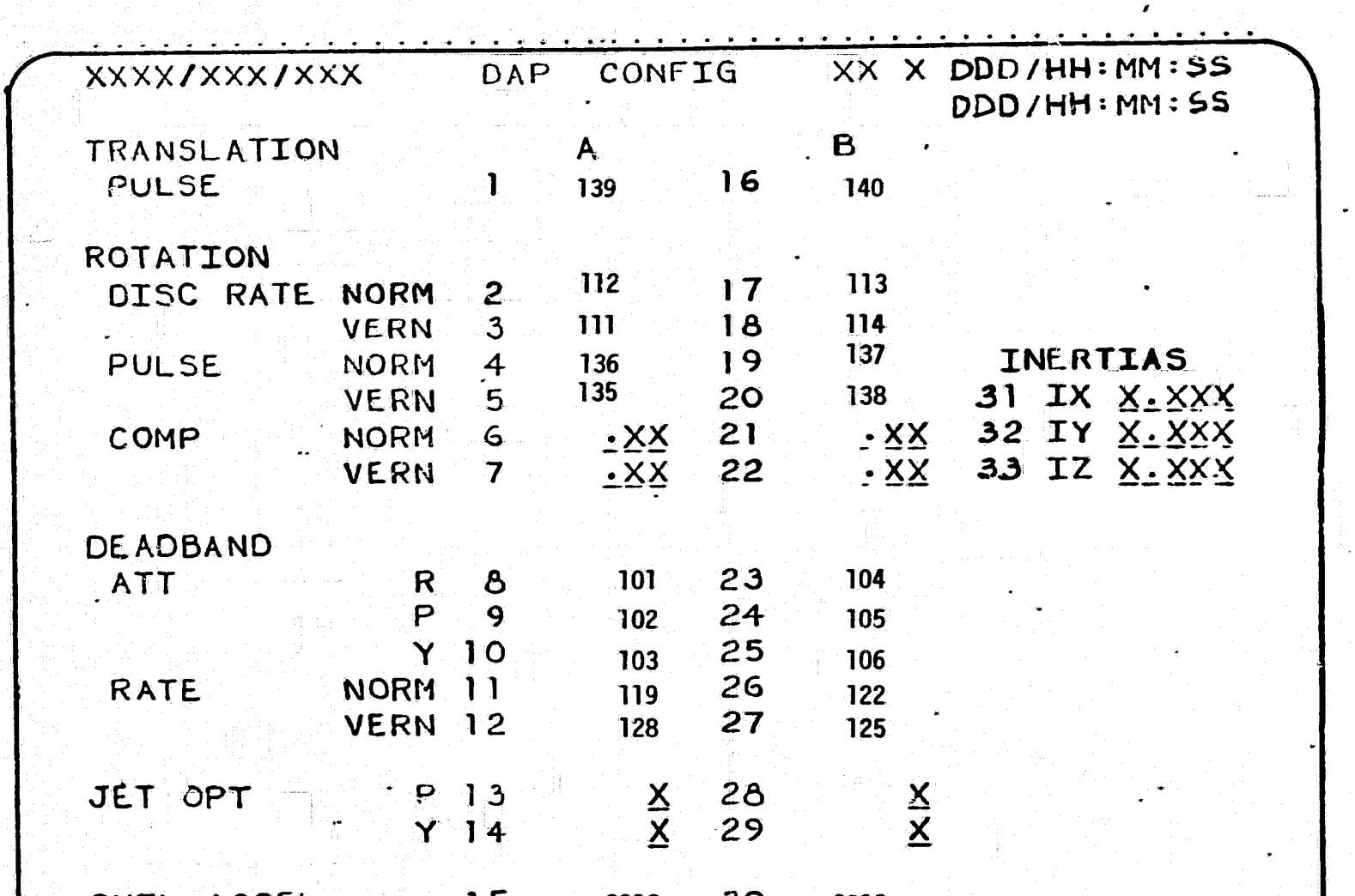


Figure 5-6.- DAP configuration spec display.

(Bold numbers denote location in FSCOM associated with the CRT input position)

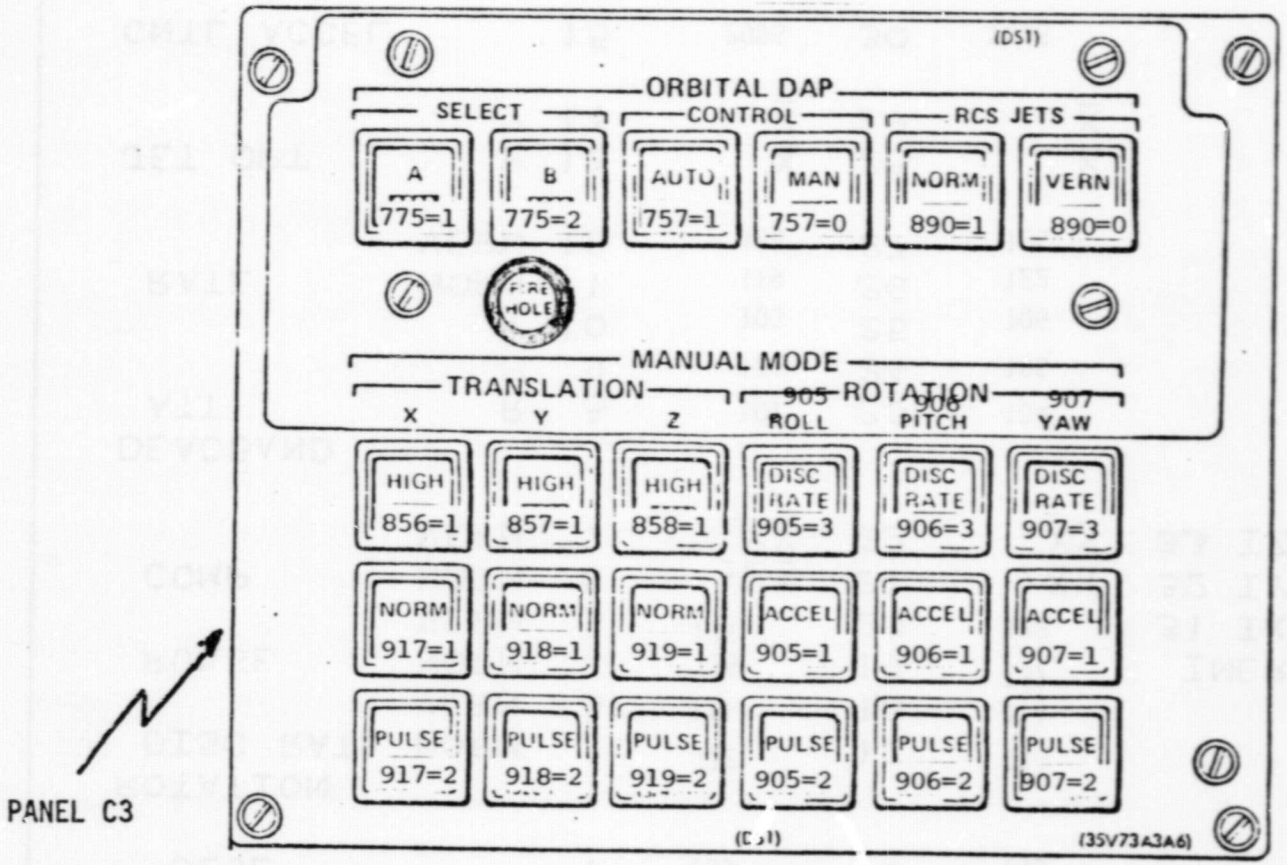


Figure 5-7.- Control panel C3 and corresponding FSCOM locations.
(Bold numbers denote location in FSCOM associated with the CRT input position)

closely to the definitions in reference 20. An extensive list of these definitions is given in the appendix.

USER INPUT REQUIREMENTS

The mission-dependent data associated with various flight control modules (Iload data) is consistent with STS-1. If the user requires changes to any of this data, the following FSCOM locations will need to be modified:

Item	FSCOM location	Jet group
Translation pulse, DAP A	139	Primary
Translation pulse, DAP B	140	Primary
Rotation discrete rate, DAP A	112	Primary
Rotation discrete rate, DAP A	111	Vernier
Rotation discrete rate, DAP B	113	Vernier
Rotation discrete rate, DAP B	114	Primary
Pulse size, DAP A	136	Primary
Pulse size, DAP A	135	Vernier
Pulse size, DAP B	137	Vernier
Pulse size, DAP B	138	Primary
Attitude deadband roll, DAP A	101	Primary, Vernier
Attitude deadband roll, DAP B	104	Primary, Vernier
Attitude deadband pitch, DAP A	102	Primary, Vernier
Attitude deadband pitch, DAP B	105	Primary, Vernier
Attitude deadband yaw, DAP A	103	Primary, Vernier
Attitude deadband yaw, DAP B	106	Primary, Vernier
Deadband rate limit, DAP A	119	Primary
Deadband rate limit, DAP B	122	Primary
Deadband rate limit, DAP A	128	Vernier
Deadband rate limit, DAP B	125	Vernier
Jet options for pitch, DAP A	115	Primary
Jet options for pitch, DAP B	116	Primary
Jet options for yaw, DAP A	117	Primary
Jet options for yaw, DAP B	118	Primary

This information corresponds to the DAP configuration specification display as shown in figure 5-6 and further explained in table 5-III. When changing the

TABLE 5-III.- DAP CONFIGURATION SPECIALIST USAGE

ITEM NO.	DESCRIPTION
1 AND 16	SIZE OF TRANSLATION PULSE IN FT./SEC. (ΔV) FOR DESIRED TRANSLATION
2 AND 17	SIZE OF DISCRETE RATE COMMAND FOR PRIMARY JETS IN DEGREES/SECOND ROTATION
3 AND 18	SIZE OF DISCRETE RATE COMMAND FOR VERNIER JETS IN FEET/SECOND
4 AND 19	THE PULSE SIZE TO ACHIEVE A DESIRED ROTATIONAL RATE PER PULSE USING THE PRIMARY JETS ONLY (IN DEGREES/SECOND)
5 AND 20	THE PULSE SIZE TO ACHIEVE A DESIRED ROTATIONAL RATE IN DEGREES/SECOND USING VERNIER JETS ONLY (RATE IS PER PULSE)
6 AND 21	ALLOWABLE ROTATION RATE (DEGREES/SECOND) FOR AXES RESULTING FROM A RATE COMMAND ABOUT ANOTHER AXIS WHEN FIRING PRIMARY JETS
7 AND 22	ALLOWABLE ROTATION RATE (DEGREES/SECOND) FOR AXES RESULTING FROM A RATE COMMAND ABOUT ANOTHER AXIS WHEN FIRING VERNIER JETS
8 AND 23	ATTITUDE DEADBAND IN DEGREES FOR ROLL (X-AXIS)
9 AND 24	ATTITUDE DEADBAND IN DEGREES FOR PITCH (Y-AXIS)
10 AND 25	ATTITUDE DEADBAND IN DEGREES FOR YAW (z-AXIS)
11 AND 26	RATE DEADBAND (ALL AXES) IN DEGREES/SECOND WHEN USING PRIMARY JETS
12 AND 27	RATE DEADBAND (ALL AXES) IN DEGREES/SECOND WHEN USING VERNIER JETS
13 AND 28	JET OPTIONS FOR PITCH: 1: FORWARD AND AFT JETS 2: FORWARD JETS ONLY 3: AFT JETS ONLY
14 AND 29	JET OPTIONS FOR YAW (CODE AS ABOVE)
15 AND 30	CONTROL ACCELERATION SELECTED FOR EITHER PRIMARY OR VERNIER JETS 0: NOMINAL 1 THRU 5: PRESTORED I-LOADED VALUES
31 THRU 33	PRINCIPAL AXIS MOMENTS OF INERTIA (X, Y, AND Z RESPECTIVELY) IN SLUGS-SQUARE FEET TIMES 10^{-6}

ILOAD data, the user must also set the following discretes if the data changes are made during (rather than at the start of) the simulation.

FSCOM(773) = 1

This is the DAP load change discrete flag which will cause the new ILOAD data to be incorporated into the appropriate flight control modules.

The following data are initialization default values for the flight control software:

Item	FSCOM location	Default value
DAP load A	775	1
Roll discrete submode	905	3
Pitch discrete submode	906	3
Yaw discrete submode	907	3
Primary RCS	890	1
X-pulse submode	917	2
Y-pulse submode	918	2
Z-pulse submode	919	1
Manual mode	757	0

This data corresponse to the panel "switches" shown in figure 5-7. To change any of these "switches" the following locations must be set:

Item	FSCOM location	Value
Translation high Z	856	1
X normal submode	917	1
Y normal submode	918	1
Z normal submode	919	1
Roll acceleration submode	905	1
Pitch acceleration submode	906	1
Yaw acceleration submode	907	1
Roll pulse submode	905	2

Item	FSCOM location	Value
Pitch pulse submode	906	2
Yaw pulse submode	907	2
Auto mode	757	1
Vernier RCS	890	0
DAP load B	775	2

When switching from either primary to vernier or vernier to primary, the following discrete must also be set:

FSCOM(774) = 1

This will cause the ILOAD data associated with that jet group to be used in the software.

Figure 5-8 depicts the OPS-2 orbit display associated with the Universal Pointing Processor. Referring to figure 5-8, the following user inputs are required in order to execute guidance maneuvers:

Item	FSCOM location	Value
Required inertial roll attitude	1891	deg
Required inertial pitch attitude	1861	deg
Required inertial yaw attitude	1900	deg
Desired pitch body vector	1847	deg
Desired yaw body vector	1849	deg
Omicron	1859	deg
Maneuver task (MNVR)	1857	1
Maneuver task	1858	1
LVLH task (LVLH)	1857	2
LVLH task	1853	1
LVLH task	1855	1
Rotation task (ROT)	1857	3
Rotation task	1853	1
Stop/attitude hold task	1857	3

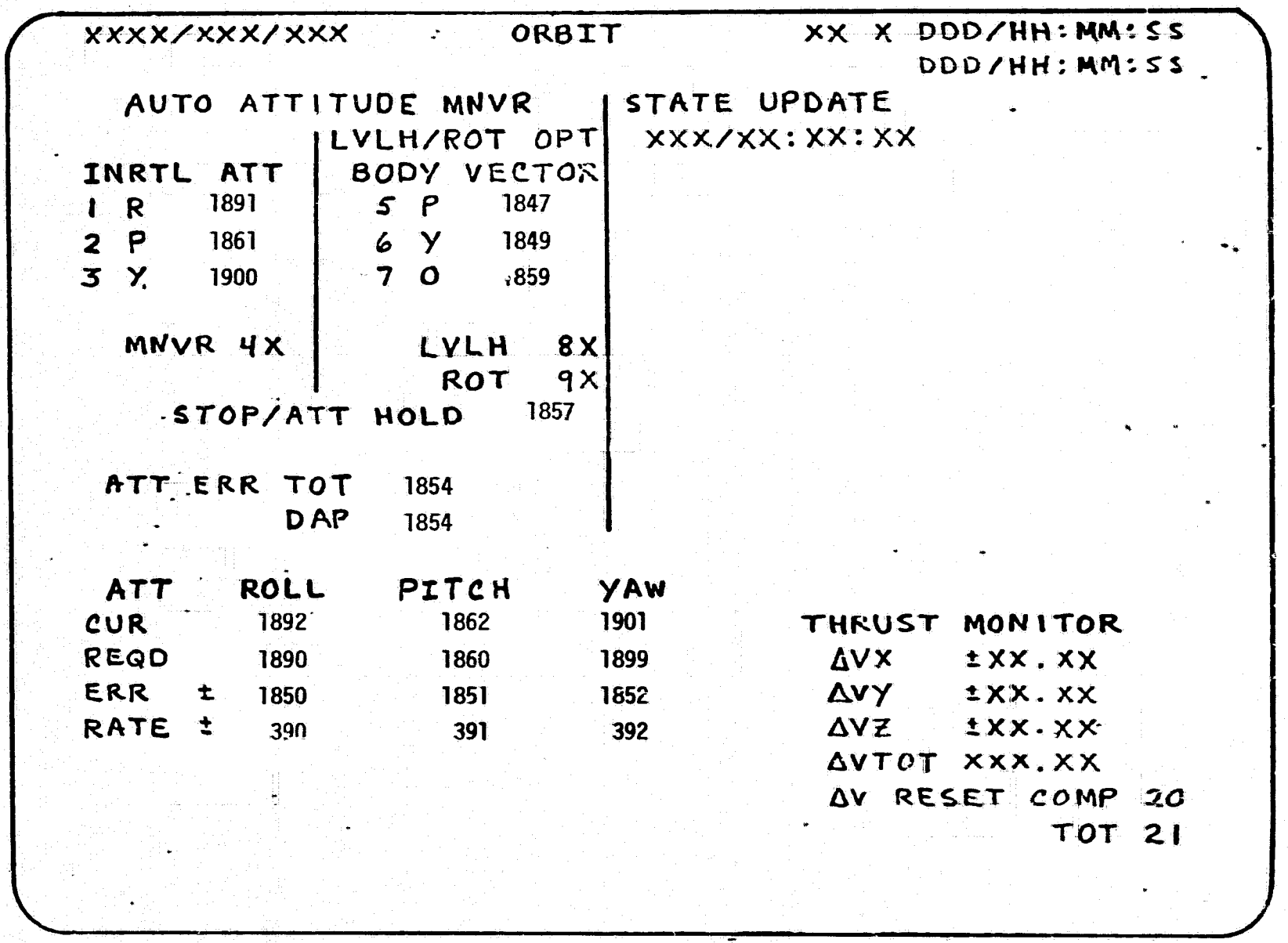


Figure 5-8.- OPS-2 orbit display.

Item	FSCOM location	Value
Total attitude error	1854	1
Total DAP error	1854	0
Current roll attitude	1892	deg
Current pitch attitude	1862	deg
Current yaw attitude	1901	deg
Required roll attitude	1890	deg
Required pitch attitude	1860	deg
Required yaw attitude	1899	deg
Total error roll	1850	deg
Total error pitch	1851	deg
Total error yaw	1852	deg
Roll rate	390	deg
Pitch rate	391	deg
Yaw rate	392	deg

The rotational hand controller (RHC) and translational hand controller (THC) inputs are required as follows:

Item	FSCOM location	Value
Roll RHC	394	±1 for on, 0 for off
Pitch RHC	395	±1 for on, 0 for off
Yaw RHC	396	±1 for on, 0 for off
X THC	413	±1 for on, 0 for off
Y THC	414	±1 for on, 0 for off
Z THC	415	±1 for on, 0 for off

The RHC defaults to the discrete rate mode and the translational THC default to the pulse mode in all areas. The area must select the primary or vernier jets (default is primary); however, the verniers are incompatable with the translation manual modes. RCS rotational mode is selected on an axis-by-axis basis (see fig. 5-7) with mixing of modes allowed. If the RHC exceeds

"soft stop" and the acceleration mode was not selected, the mode selected becomes active (on axis involved) when RHC returned to detent. To exceed the soft stop, the following location must be set:

Item	Location	Value
Roll RHC soft stop	406	1 to exceed, 0 not
Pitch RHC soft stop	407	1 to exceed, 0 not
Yaw RHC soft stop	408	1 to exceed, 0 not

Free drift is in effect after the "soft stop" when returning to detent with attitude hold in detent. If the acceleration mode was selected, acceleration command is effected when the RHC is out of detent with free drift when in detent. If in the automatic mode and a RHC command is effected, then the manual mode will be automatically selected.

6. REFERENCES

1. Davis, J. P.: Space Shuttle Functional Simulator, Vol. I - System. JSC-06726, Rev. C; LEC-8480, Rev. C; Dec. 1979.

2. Bald, N. B.: Space Shuttle Functional Simulator, Vol. II - User's Guide. JSC-06726, Rev. D; LEC-8480, Rev. D; Sept. 1979.

3. Phillips, Robert F., II: Space Shuttle Functional Simulator, Vol. IV - Onorbit Environment. JSC-16902, LEMSCO-15350, July 1980.

4. Bald, N. B.: SSFS Utility Routine Documentation Series - EXTRCT. LEC-1267, Oct. 1973.

5. American Ephemeris and Nautical Almanac for the Year 1980. U.S. Government Printing Office, Washington, D.C., 1979.

6. Bate, R. R.; Mueller, D. D.; White, J. E.: Fundamentals of Astrodynamics. Dover Pub., New York, 1971.

7. Astronomical Almanac for 1981. U.S. Government Printing Office, Washington, D.C., 1980.

8. Underwood, J. M.: Orbital Aerodynamics. JSC Memo EX33, August 14, 1975.

9. Babb, G. R., and Bean, W. C.: Analytical Atmospheric Density Model. JSC Memo FM5 (75-4), Jan. 27, 1975.

10. Duhen, D. D.: Proposed Changes to the Space Shuttle On-Orbit Aerodynamic Model for the SSFS Computer Program. MDTSCO Design Note 1.4-2-15, Dec. 17, 1975.

11. Kempen, E. E.: Gravitational Potential Models. 672-21-PA-261, Oct. 1969.

12. Space Operational Data Book, Vol. II, Mission Mass Properties, Revision A. Sept. 1975, Amendment 4, Jan. 28, 1976.

13. Aerodynamic Design Data Book, Vol. I, Orbiter Vehicle. Rockwell International, Space Division, SD72-SH-0060-1I.

14. FD2 Configuration Description. Rockwell International, Space Division, RI Internal Letter, SAS/FSA/74/164, April 29, 1975.

15. Space Shuttle Flight Control System Data Book, Vol. II - Orbiter. Rockwell International, Space Division, SD73-SH-0097-2E, June 1976.

16. Orbiter Aerodynamic Design Data Book. Rockwell International, Space Division, SD72-SH-0060-1H, Rev. 3, May 5, 1975.

17. Parris, K. M.: SSFS Utility Routine Documentation Series - POLY. LEC-3324, April 1974.

18. Parris, K. M.: SSFS Utility Routine Documentation Series - PEVAL. LEC-3323, April 1974.

19. Manned Maneuvering Unit Design Data Book, MMU-SE-11-01, Dec. 16, 1979.

20. Functional Subsystem Software Requirements Document, Space Shuttle Orbital Flight Test, Level C, Guidance, Navigation, and Control, Part C, Flight Control, Onorbit DAP 1. SD-76-SH-0094, Nov. 1979.

21. Orbital Flight Test - Insertion/Onorbit/Deorbit Guidance, Navigation and Cotnrol. I/O/D2101 to 2401, JSC Flight Training Branch, April 24, 1978.

APPENDIX
FSCOM LOCATIONS

Program symbol	Location	Type	Dimen.	Definition
AUTO	1	L		Auto manual switch, true = auto
AXIS	2	I		Counter, set internally on 1, 2, 3
DEGTOR	3	R		Degrees to radians
DYINIT	4	I		
FTOM	5	R		Feet to meters
GTMPSS	6	R		Gravity, meters per sec^2
HOLD	7	I		Attitude hold flag
IDMAT	8	R	9	Identity matrix
INTOM	17	R		Inches to meters
LBFTON	18	R		Foot pounds to newtons
LBTOKG	19	R		Pounds to kilograms
MANUAL	20	L		
MVRTRK	21	I		
NMTOFP	22	R		Newton meters to ft lb
NUL	23	I		
OFF	24	L		= FALSE, a mnemonic
ON	25	L		= TRUE, a mnemonic
PI	27	R		3.1415
PRIMRY	30	L		
RTODEG	34	R		Radians to degrees
SLTOLB	35	R		Slugs to pounds
TDAP	36	R		Period of flight control minor cycle
TVC	38	I		Thrust vector control
TWOPI	39	R		6.28...

Program symbol	Location	Type	Dimen.	Definition
ARCOMP	89	R	12	Array compensation thresholds
ARDB	101	R	6	Attitude deadband limits RPY for dapload A then B
ARDISR	107	R	4	Attitude discrete mode rates, nominal then vernier, A then B
ARMVR	111	R	4	Maneuver rate
ARNPL	115	I	2	Nominal nose/tail jets, pitch A then B
ARNYWL	117	I	2	Nominal nose/tail jets, yaw A then B
ARRLIM	119	R	12	Rotational rate limits, RPY, nominal then vernier, A then B
ARROHL	131	R	4	Rotational rate limits, RPY, nominal then vernier, A then B
ARROPL	135	R	4	Rotational pulse level, nominal then vernier, A then B
ARTPLS	139	R	2	Translational pulse level x, y, z for dapload A then B
ATERPL	141	R	3	
ATI6HZ	144	R	3	Attitude increment, low roll rate, deg
ATTER	147	R	3	Attitude error PYR
DVBSEC	150	R	3	
ETSEP	153	L		External tank separation, not used
HFAIL	154	L	44	JET FAILURE, TRUE = failed
HFLCHG	198	L		Change in jet failure status has occurred this pass if true
IMUFLG	199	L		IMU ATT GOOD FLAG 1 = FALSE
INERTR	200	R	3	Diagonal components of vehicle inertial ratio xx, yy, zz
MCACC	243	R	3	Magnitude of control accelerations along x, y, z
MTRKOP	246	L		Maneuver track option enable, 1 = enabled
NEWJON	247	L		Turn-on jet command present for ith jet

Program symbol	Location	Type	Dimen.	Definition
PLEXT	371	I		Payload extended flag
PPACEL	372	R	3	Phase plane accelerator levels, depends on primary vernier, or the P/L extended flag
QBM50	375	R	4	M50 to body quaternion measured
QBM50C	379	R	4	M50 to body quaternion required
QBM50P	383	R	4	M50 to body quaternion display
RCMDPL	387	R	3	
REST	390	R	3	IMU derived body rate estimates about x, y, z
RESTRT	393	L		
RHC	394	R	3	Rotational hand controller state -1, +1, roll, pitch, yaw
RHCST	397	I	3	Rotational hand controller state
ROTJC	400	R	3	RCS rotation command
SFTSTP	406	L	3	Soft stop, status of RHS, each axis
TCMD	412	R		GMT of required body quaternion
THC	413	I	3	Translational hand controller commands x, y, z, -1, +1
TRCMD	418	R	3	x, y, z components of required body attitude rate
VMASS	422	R		Vehicle mass
AGPRI	425	R		Acceleration gain, primary
AGVER	427	R		Acceleration gain, vernier
AG1PRI	428	R		Attitude gain 1, primary
AG1VER	430	R		Attitude gain 1, vernier
AG2PRI	431	P		Attitude gain 2, primary
AG2VER	433	R		Attitude gain 2, vernier
AMM	434	R		Approximate/exact algorithm threshold
RG1PRI	468	R		Rate gain 1 primary
RG1VER	470	R		Rate gain 1 vernier
RG2PRI	471	R		Rate gain 2 primary
RG2VER	473	R		Rate gain 2 vernier

Program symbol	Location	Type	Dimen.	Definition
RLIM	475	R		Phase phase rate limit
TRNINC	480	R	42	Translational increments 14 jets, x y z for each
T1	522	R		Threshold for use of second vernier jet
T2	523	R		Threshold for use of third vernier jet
WFRATE	524	R		Scale factor for off-axis vernier preference
ANGINC	530	R	150	Angle increments 26 jets, x, y, z for each 0.08 sec firing
JETMAP	680	R	38	Jet numbers
MCAALT	718	R	15	Magnitude of control acceleration vernier alternate
PRIMMI	733	R		Rotational rate change due to minimum impluse burn with primary jet
PTRM1E	734	R		
TNBBOD	735	R	9	Transformation to body coordinates
TRANMI	744	R		
VERNMI	745	R		Largest rotational rate change due to minimum impulse burn with vernier jet
AEST1	748	R	3	Attitude estimate 1
AEST2	751	R	3	Attitude estimate 2
AGAIN	754	R		Acceleration gain chosen
AGAIN1	755	R		Attitude gain 1 chosen
AGAIN2	756	R		Attitude gain 2 chosen
ATOMAN	757	L		Automatic or manual, AUTO = TRUE
ATT	758	R	3	Attitude degrees
ATTHLD	761	L		Att hold flag
BYPASS	766	L	13	Bypass this axis
CLCNTR	769	I		Counter in RECON

Program symbol	Location	Type	Dimen.	Definition
DAPCHG	773	L	2	Change in dapload present, first element true for change, second element true for primary/vernier
DAPLD	775	I		Dapload index - 1 = dapload A, 2 = dapload B
DATINC	776	R	3	Desired attitude increment
DATT	779	R	3	Desired attitude
DB	782	R	3	Deadband to phase plane
DBREQ	785	R	3	Deadband requested
DBRT	788	R	3	Desired body rate
DISRT	791	R		
DVMIMP	795	R	3	Change in body angle rate due to minimum impulse
DWRCS	798	R	3	Rate change due to jet firing
FLG6HZ	849	L		
FORFIR	850	L	3	Rate damping flags
HILOZ	856	L		Manual mode translation, z high regardless of T or F
IAMTRK	857	I		Initiate auto maneuver track
INITJS	860	L		Initiate jet select flag
IP1FTR	861	L		Initiate part 1 filter flag
IRCSER	862	L	3	Initiate RCS errors flag
IRDISC	865	L		Initiate rotation discrete flag
IRPLSE	866	L		Initiate rotation pulse
ITPLSE	867	L		Initiate translation pulse
LOPTN	871	L		Tail of nose jet selection for low pitch rotation level, 1 = tail 0 = nose
LOYWTN	872	L		Tail or nose jet selection for low yaw rotation
LROTOP	873	I	3	Last rotation option
MAGMVR	876	R		Magnitude of auto maneuver rate

Program symbol	Location	Type	Dimen.	Definition
PVINDX	889	I		Primary or vernier jet index, 1 = P, 2 = V, depends on PVSW
PVSW	890	L		Primary or vernier jet switch, true = normal
REST2	895	R	3	Rate estimate 2
RGAIN1	898	R		Rate gain 1
RGAIN2	899	R		Rate gain 2
RLAMP	900	I	3	Rotation underway lamp
ROHILO	903	L	2	Pitch then yaw acceleration level selection 1 = nominal
ROTOPT	905	I	3	Manual mode rotation, R, P, Y 1 = accel, 2 = pulse, 3 = discrete
RPLSES	908	R		
RTER	909	R	3	Attitude rate error
RTLIM	912	R	3	Attitude rate limit
TOPT	917	I	3	Manual translation mode for x, y, z 1 = normal, 2 = pulsed
TPLSES	920	R		Translational pulse size, depends on deadband
TRANJC	921	I	3	Translational jet command
UNDSAC	930	R	3	Undesired acceleration, each axis
AAOPT	933	I		Auto maneuver option, 1 = hold, 2 = maneuver
FCCNTR	940	I		Counter in RECON
OLAAOP	946	I		Last pass auto maneuver option
OLBYPS	947	L	3	Last pass bypass phase plane flag
OLDAM	950	L		Last pass auto maneuver change flag
OLDTOP	955	L	3	Last pass translation HC option
OLMODE	958	I		Last pass auto or manual mode
OLROPT	960	I	3	Last pass rotation HC option
OLSSTP	963	L	3	Last pass soft stop flag
AM	976	R		Auto maneuver eigen angle
BIAS	977	R		Auto maneuver bias in phase plane

Program symbol	Location	Type	Dimen.	Definition
BIASV	978	R	3	Auto maneuver bias vector
DELTA	981	R	3	Not used
MTRACK	986	L		Auto maneuver flag, T = auto, F = hold
OLDMT	987	L		Last pass auto maneuver flag
OMGCMD	988	R	3	Commanded auto maneuver rate
QCB	991	R	4	Commanded quaternion
QGMD	995	R	4	Commanded quaternion
VR	999	R	3	Auto maneuver rotation vector
WEST	1002	R		Auto maneuver parameter
AZANG	1006	R		Compensated azimuth gimbal angle
CDP	1007	R		Cosine 1/2 DP
COSP	1008	R		Cosine of pitch
COSR	1009	R		Cosine of roll
COSY	1010	R		Cosine of yaw
CP	1011	R		Cosine of gimbal angle 3
CRMN	1012	R		Cosine of gimbal angle 2
CRPL	1013	R		Cosine of gimbal angle 1
CTR	1014	R		Inner loop, outer loop flag
CY	1015	R		Cosine of gimbal angle 4
DELGA	1016	R	4	Change in gimbal angles
DP	1020	R		IMU roll/pitch nonorthogonality
GA	1021	R	4	Gimbal angles
GAPREV	1025	R	4	Previous gimbal angles
GAREAD	1029	R	4	Gimbal angles relative to ADI
GATOBD	1033	R	4	Gimbal angles to body
IATPRC	1045	L		Initiate attitude processor flag
IRANG	1047	R		Compensated inner roll gimbal angle
MROLLB	1061	R	9	IMU roll to body system transformation
OLDSIN	1070	R		Holder for old gimbal angle sines

Program symbol	Location	Type	Dimen.	Definition
ORANG	1071	R		Compensated outer roll gimbal angle
PANG	1072	R		Compensated pitch gimbal angle
QBPREV	1073	R	4	Previous quaternion body to M50
QBROLL	1077	R	4	Body roll wrt gimbal angle quaternion
QROLSM	1081	R	4	Roll wrt stable member quaternion
QSMM50	1085	R	4	Stable member to M50 quaternion
QUPD	1089	R	4	Update quaternion
SDP	1093	R		Sine 1/2 DP
SINP	1094	R		Sine of pitch
SINR	1095	R		Sine of roll
SINY	1096	R		Sine of yaw
SP	1097	R		Sine of gimbal angle 3
SRMN	1098	R		Sine of gimbal angle 2
SRPL	1099	R		Sine of gimbal angle 1
SY	1100	R		Sine of gimbal angle 4
TCLM50	1101	R	9	Transform cluster to M50
TNBROL	1110	R	9	Transformation, roll to body
TNOW	1116	R		Time now, sec
ARES1	1119	R	3	Rate estimate 1
ARES2	1122	R	3	Rate estimate 2
ATTINC	1125	R	3	Attitude increment
DTOMGA	1128	R	3	Rate estimate variable (local)
MSDATT	1131	R	3	Rate estimate variable (local)

Program symbol	Location	Type	Dimen.	Definition
TMEAS	1134	R		Time
TMSSQ	1135	R		DAP time squared
C	1136	R		
DELTHA	1137	R	3	
DRSTRT	1138	L		
JETCMD	1141	R		
OLDDR	1142	L	3	Local variables in phase plane
OLDFF	1145	L	3	
SUD	1148	R		
S11	1149	R		
TEMP	1150	R		
XXIS	1151	I		
X1	1152	R		
X2	1153	R		
CHECK1	1174	L		
CHECK2	1175	L		Two jet failures in nose group if true
CHECK3	1176	L		
CHECK4	1177	L		
CHECK5	1178	L	4	
CHECK6	1182	L	4	
CHECK7	1186	L		
CHECK8	1187	L		
COMCMD	1188	R	3	

Program symbol	Location	Type	Dimen.	Definition
DVRCS	1191	R	3	
DYFAIL	1194	L	2	Denotes failure of both y jets for a forward cluster if true
DZFAIL	1196	L		
DZFL2	1197	L		Denotes two failed -z jets on a side if true
HFLJSL	1204	L	38	
HIPIT	1242	L		High pitch, true = high rate
HIROLL	1243	L		High roll, true = high rate
HIX	1244	L		High x rate, true = high rate
HIYAW	1245	L		High yaw, true = high rate
HIZ	1246	L		High z translation command present
I	1247	R		
J	1248	R		
JETTAB	1249	I	38	Divides jets into 14 bunches
JONJSL	1287	L	38	
K	1325	I		
L	1326	I		Local variables jet select logic (JSL)
NODUMP	1327	L		
NOM	1328	L		Nominal
NOSECK	1329	L	6	Nose check z, true = low nose rotation, false see TAILCK
OLDJON	1330	L	6	
OLDRJC	1336	I	3	
RACCUM	1358	R	3	
RCSDMP	1361	L		
ROTMI	1362	R	3	
ROTVAL	1363	R		Local variables in JSL
SUMDV	1366	R	3	
SZFAIL	1369	L	3	
SZFL	1372	L		

Program symbol	Location	Type	Dimen.	Definition
TACCUM	1373	R	6	
TAILCK	1379	L		Tail check z, true = low rotation tail, else, see FSSR
THRENZ	1380	L		
TWOZFL	1381	L		
VFAIL	1382	L		Denotes failure(s) exist(s) in vernier jets
VRNCMD	1383	R	3	
XFEED	1386	L		Crossfeed interconnect flag, true = crossfeed open
XFMINV	1387	R	3	
ZFAIL	1390	L	5	
OLDTHC	1398	I	3	Old translation HC command
ORHCST	1404	I	3	Old rotational HC command
FL	1603	L	46	
JETFLG	1649	I	2	Variables that interface FSW with RCS model
JPFLG	1651	I		
P5501	1652	R	46	
RFLG	1698	I	3	Region flag in phase plane
SSFSNO	1701	I	44	RCS jet numbers for RCS model
ANG	1836	R	3	Current inertial angles
ARBBCS	1839	R		
ATTERR	1840	R	3	Attitude error
ATTRAT	1843	R	3	Rate error
BODVP	1846	R		
BODVPD	1847	R		Pitch component of body, desired
BODVY	1848	R		
BODVYD	1849	R		Yaw component of body, desired, deg
ERR	1850	R	3	Total error in degrees, R, P, Y
IBDVD	1853	R		Total attitude error
IERSEL	1854	R		Error select flag

Program symbol	Location	Type	Dimen.	Definition
IOMID	1855	R		Task flag 1 = LVLH
IOPACT	1856	R		1 roll or yaw, auto maneuver, 2 = WLH alignment
IOPSEL	1857	R		Task flag, 1 = MNVR, 2 = LVLH, 3 = RPT
MNVD	1858	R		Maneuver option discrete
OMICRN	1859	R		Omicron
PITCHC	1860	R		Required pitch inertial
PITCHI	1861	R		Initial input for pitch (ADI)
PITCHN	1862	R		Current pitch
PLOS	1863	R	3	Desired rotation vector
QADIM	1866	R	4	Quaternion ADI to inertial
QBADI	1870	R	4	Quaternion body to ADI
QBADIC	1874	R	4	Quaternion body to ADI commanded
QBBC	1878	R	4	Quaternion body to body commanded
QMADI	1882	R	4	Inertial to ADI quaternion
QMNVR	1886	R	4	Commanded quaternion
ROLLC	1890	R		Required (roll, inertial)
ROLLI	1891	R		Initial input for roll (ADI)
ROLLN	1892	R		Current roll
RVMAG	1893	R	3	Attitude error magnitude
TOTERR	1894	R	3	Total error
YAWC	1899	R		Required yaw, inertial, deg
YAWI	1900	R		Yaw required, inertial, deg (ADI)
YAWN	1901	R		Current yaw
EULER	2055	R	3	Euler angle array
CNTACA	2073	I		Control acceleration index
CNTACB	2074	I		Control acceleration index

Program symbol	Location	Type	Dimen.	Definition
MCACCP	2075	R	3	Mean acceleration level, primary
MCACCV	2078	R	3	Mean acceleration levels, vernier
MCACPL	2081	R		Mean acceleration levels with payload extended
CNTLACC	2096	I		Control acceleration index = CNTACA for dapload A = CNTACB for dapload B
DLIM	2172	R		1° tolerance factor
DOTT	2173	R		Local variable (dot product)
EA	2174	R		Angle of rotation
INIOPT	2176	I		Task flag 1 = LVLH, 2 = rotation
INPASS	2177	I		Initial pass flag
IOPPRC	2178	I		Task flag 1 = LVLH, 2 = rotation
MTP	2179	R	9	Transformation matrix M50 to required body
MTT	2188	R	9	Local variable matrix
DNINTY	2197	R		90°
PLOSA	2198	R	3	Desired pointing vector
PTM	2201	R	9	Required body to M50 transformation
QATM50	2210	R	4	Commanded quaternion
QBAM50	2214	R	4	Commanded quaternion
QMNVRA	2218	R	4	Commanded quaternion
QRBB	2222	R	4	Local variable
RAVGAB	2226	R		Magnitude of position vector
ROLL	2227	R		90°
RRABOD	2228	R	3	Unit vector on y-axis
RRATA	2231	R		Desired maneuver rate
RRBOD	2232	R	3	Local variable
RRM50	2235	R	3	Local variable
RXVAVG	2238	R	3	Local variable
SEA	2241	R		Local variable

C-4

Program symbol	Location	Type	Dimen.	Definition
TLOS	2142	R	3	Unit position vector
TLXVAV	2245	R	3	Local variable
UVMXRM	2248	R	3	Local variable
VBXRRB	2251	R	3	Local variable
VBXYN	2254	R	3	Local variable
VECBOD	2257	R	3	Desired pointing vector
VECM50	2260	R	3	Unit position vector
VMXRRM	2263	R	3	Local variable
VMXYT	2266	R	3	Local variable
VVMXRM	2269	R	3	Local variable
WBI	2272	R	3	Inertial rotation rate
YN	2275	R	3	Local variable
YT	2278	R	3	Local variable
YVEC	2281	R	3	Unit vector along Y(= 0, 1, 0)
ZVEC	2284	R	3	Unit vector along Z(= 0, 0, 1)
SRAVGG	2287	R	3	Shuttle position M50, inertial single precision
SVAVGG	2290	R	3	Shuttle velocity, M50, inertial single precision
UVVXRM	2293	R	3	Local variable
CROLL	2296	R		Cosine of roll
SROLL	2297	R	3	Sine of roll
SVAVGN	2298	R	3	Unit vector of velocity